The Quest for a General Theory of Robustness in Learning Machines

... with Fisher Information Metric/Matrix, Information Geometry ...
... Integrable Foliation, Frobenius Theorem, Leaves, Riemannian curvature ...
... then Sequential models, Transformers, Recurrent Neural Networks, Graph Neural Networks ...
... Uncertainty quantification, Extended Variational Density Propagation, MNIST, CIFAR ...
... Normal Distribution, Stochastic Differential Equations, Gaussian Process ... *something something* ... Takens Theorem ...
... Trajectory Prediction, Applications, Complexity Prediction and Mitigation, TAS and Mass estimation using Wind and Temperature, Metaheuristics ...
... Air Traffic Management, OpenSky, BlueSky, En-route, 40 minutes, environmental constraints ... Extended ATC Planner, CRNA-Est ... DGAC .....

In this document, I gather all the leads I found concerning the robustness of neural networks to adversarial attacks. The goal is to find interesting conjectures that could be explored theoretically or tested experimentally. Next, I should focus on sequential models for regression, since papers about adversarial attacks tend to focus on classification. We may propose a defense against the attack of Tan et al. [1]

Adversarial attacks were first discovered for deep learning by Szegedy et al. [2] in 2013 (adversarial attacks for linear classifiers were discovered in 2004). It has been shown experimentally that adversarial attacks are:

- Ubiquitous. Given a state-of-the-art model trained in a standard manner, every point of the training set has a adversarial example close to it.
- Transferable. Adversarial examples crafted using one model tend to work on models with completely different architecture provided that they have been trained on the same task. However, we will see that it is possible to craft adversarial examples that are not transferable.

Adversarial attacks are often classified along several criteria:

- **White-box attacks** vs. **Black box attacks** $\rightarrow$ can the attacker access the model's parameters? (threat model)
- **Untargeted attacks** vs. **Targeted attacks** $\rightarrow$ does the attacker want the adversarial example to be classified as a specific target class, or simply to be misclassified? (objective)
- Choice of the **distance metric**, generally a l_p norm $\|x - \tilde{x}\|_p < \epsilon$ including the l_0 "norm". Some works also investigate unconstrained attacks.

Note that I am focusing here on **evasion attacks** i.e., attacks performed on a trained model during inference. Adversarial attacks are not the only possible evasion attacks, for example there is also *adversarial reprogramming* where a trained model can be used for another purpose. Many other attack strategies are possible depending on the goal, knowledge, and access capabilities of the attacker. For example, let us mention:

- **Data poisoning**. The training set is modified to change the behavior of the model. This strategy focus on the training part rather than the inference part. Our framework may provide insights for this attack strategy too. Examples of data poisoning are: backdoor attacks, Trojan attacks.
- **Byzantine attacks**. This strategy is concerned with distributed models, for example in *federated learning*, where a minority of malicious participants can hack the model.

- **Model extraction**. Here, the goal of the attacker is *espionage*. The attacker wants to retrieve the training data or the model's parameters without any access to the model (black box). This is linked with privacy issues. Examples of model extraction attacks are: model stealing, membership inference, model inversion.

Contents

1 Adversarial Attacks

In this section, I briefly describe the most well known adversarial attacks methods. I would then like to classify them according to their fundamental ideas for crafting adversarial attack.

1.1 One-step gradient attacks

It seems that one-step attacks are more transferable than iterative attacks (see the reference in Xu et al. review [3] §5.3).

1.1.1 Fast Gradient (Sign) Method

Goodfellow et al. [4] 2015, Kurakin et al. [5] 2016. There is also a momentum version of FGSM called MI-FGSM.

1.1.2 One-Step Target Class Method

Kurakin et al. [5] 2016.

1.1.3 Jacobian saliency maps

See Papernot et al., The limitations of deep learning in adversarial settings, 2016.

1.2 Iterative attacks

1.2.1 Basic Iterative Method

Kurakin et al. [5] 2016.

1.2.2 Projected Gradient Descent

Madry et al. [6] 2018.

1.2.3 DeepFool

Moosavi-Dezfooli et al. [7] 2016.

For now, I have only read summaries of the paper. DeepFool seems to be an unrestricted attack (the only parameter is the number of iteration). The idea is to start with a linear classifier for binary classification. There is an easy formula to obtain the distance of any point to the decision boundary (which is an hyperplane). For a nonlinear model, you simply compute the gradient to have a linear approximation of the model around the current iterate and use the same formula as in the linear case to make a step towards (hopefully) the decision boundary. For several classes, you select the n classes with highest probability (except the actual class) and make a step towards the closest class (according to the linear approximation).

1.3 Optimization attacks

1.3.1 L-BFGS

Szegedy et al. [2] 2013.

1.3.2 Carlini and Wagner attack

1.4 Unrestricted attacks

For example: adversarial rotations, translations, and deformations. See spatial attacks in Xiao et al. [8].

1.5 Invariance attacks

In [9], the authors introduces a new typology of attacks. They call *sensitivity attacks* every attack where a small perturbation changes the prediction of the model while the true label hasn't changed. All other attacks mentioned in this section are sensitivity attacks. The author introduces *invariance attacks* where a small perturbation changes the true label while the prediction of the model doesn't change.

I haven't read the paper but the authors seem to claim that there is a trade-off between sensitivity robustness and invariance robustness, which makes sense intuitively. I'm still not sure if and how we should take into account these invariance vulnerability. When addressing sensitvity attacks, we assume that the true label doesn't change in a small (or not so small) neighborhood of each training and test points. This is in contradiction with invariance attacks, and it is certainly not true as we can see in some images of [9] where a small change in l_2 norm leads to a semantically different image. This is coherent with the idea that pixel-based coordinates and l_p norm are very bad measures of "distinguishability". However, the problem seems far more difficult than sensitivity attacks since it is lined to the "semantic content" of the task (when does the true label change?) and thus to the learning algorithm. Once again, it seems to show that gradient descent is unable to learn the "true semantic content" of the task, but only statistical correlation that may or may not be linked with the semantic content. This has been already discussed in [10].

1.6 Black-box attacks

1.6.1 Zeroth-order attacks

1.6.2 Query-based attacks

1.6.3 Transfer-based attacks

1.7 Spectral attacks

1.7.1 One-Step Spectral Attack

Zhao et al. [11] 2019.

1.7.2 Dog-leg Attack

Tron et al. [12] 2022.

2 Adversarial Defenses

In this section, I briefly mention the most classical defenses against adversarial attacks. Here, I use the taxonomy:

- Gradient masking
- Robust optimization
- Adversarial detection

Another possible taxonomy is:

- Modifying data / Modifying input for perturbation removal
- Modifying model
- Adding external/auxiliary tools

2.1 Gradient masking/obfuscation

The main weakness of the gradient masking strategy is that it can only "confound" the adversaries; it cannot eliminate the existence of adversarial examples.

2.1.1 Defense distillation

Some works mentioned in [13] try to train a new model to mimic another already trained model, adding some regularization terms to the loss in order to encourage the two models to be similar. It is possible to distill a robust model using these methods. The work in [13] can be seen as a "data distillation" instead of a model distillation.

In fact, the defense distillation seems to be a little bit different to the model distillation in other context. The idea is to train a first model with one-hot encoding. This model will be accurate but not robust. Then, you train a second model to predict the predictions of the first model (called the "teacher model"). Since the first model will not predict one-hot vectors, there is a kind of smoothing, or randomization, that helps the second model to be more robust. There seems to be even more sophistication to it. Let N_T be the function that outputs the *logits* of the teacher model (i.e., just before the softmax). We compute a *soft label* $\hat{y}$ as:

$$\hat{y} = \sigma(N_T(x)/T),$$

where σ is the softmax function, and T is an hyperparameter called the *temperature*. If $N(x)$ are the logits of the distilled model currently in training, the loss function is:

$$l(\hat{y}, x) = -\sum_{i=1}^{c} \hat{y}_i \log(\sigma(N(x))_i).$$

2.1.2 Shattered gradient

Add a non-differentiable pre-processing such that the model is no longer differentiable with respect to the inputs.

It is also possible to prevent transferability, see NULL labeling.

2.1.3 Stochastic/randomized gradients

Train several models and use one of them randomly. Or drop randomly some neurons. Or transform the input randomly. See data compression/randomization, feature squeezing. See mask layer.

2.1.4 Exploding & vanishing gradients / Denoising

Use a generative model to project the attack to the "data manifold" (whatever that means) then feed the projected point into the model. The concatenation of the generative model and the original model creates a very deep network such that the gradients with respect to the input tend to vanish/explode, and hence are hardly usable to craft adversarial attacks.

Examples: Defense-GAN, PixelDefend.

See also the suppression of adversarial noise with Deep Contractive Networks. See also High-Level Representation Guided Denoiser.

2.2 Robust optimization

Two possibilities:

- Minimize the average adversarial loss (the maximal loss in a neighborhood of each training point).
- Maximize the average minimal perturbation distance.

These methods focus on l_p norms perturbation. But there are other types of attacks e.g., spatial attack [8].

2.2.1 Regularization methods

Parseval networks, Cisse et al. 2017 [14] In one sentence, the idea is to penalize the Lipschitz constant of each layer. I my opinion, the paper is badly written so it is sometimes difficult to understand what they are really doing. Nonetheless, their method seems somewhat close to mine. However, the justifications for the method seem very different.

Define the two generalization errors:

$$\begin{aligned} L(W) &= \mathbb{E}_{(x,y)\sim\mathcal{D}}[l(g(x,W),y)], \\ L_{adv}(W,p,\epsilon) &= \mathbb{E}_{(x,y)\sim\mathcal{D}}[\max_{\tilde{x}:\|\tilde{x}-x\|_p\leq\epsilon} l(g(\tilde{x},W),y)], \end{aligned}$$

where $\mathcal{D}$ is the distribution of the data, g is the model, W are the weights of the model, l is the loss function (typically the log-likelihood). By definition, we have $L(W) \leq L_{adv}(W,p,\epsilon)$ for every p and $\epsilon > 0$. To achieve robustness, we want to upper bound $L_{adv}(W,p,\epsilon)$.

Assume that, for any W and any y, both the loss function l and the model g are Lipschitz continuous with respect to their input:

$$\begin{aligned} \forall p, \exists \lambda_p, \forall y, \forall z, z', |l(z,y) - l(z',y)| \leq \lambda_p \|z - z'\|_p, \\ \forall p, \exists \Lambda_p, \forall W, \forall x, x', \|g(x,W) - g(x',W)\|_p \leq \Lambda_p \|x - x'\|_p. \end{aligned}$$

Let $(x,y) \sim D$. We have:

$$\begin{aligned} \max_{\tilde{x}:\|\tilde{x}-x\|_p\leq\epsilon} l(g(\tilde{x},W),y) &= l(g(x,W),y) + \max_{\tilde{x}:\|\tilde{x}-x\|_p\leq\epsilon} l(g(\tilde{x},W),y) - l(g(x,W),y), \\ &\leq l(g(x,W),y) + \left| \max_{\tilde{x}:\|\tilde{x}-x\|_p\leq\epsilon} l(g(\tilde{x},W),y) - l(g(x,W),y) \right|, \\ &\leq l(g(x,W),y) + \max_{\tilde{x}:\|\tilde{x}-x\|_p\leq\epsilon} |l(g(\tilde{x},W),y) - l(g(x,W),y)|. \end{aligned}$$

Taking expectation:

$$\begin{aligned} L_{adv}(W,p,\epsilon) &\leq L(W) + \mathbb{E}_{(x,y)\sim\mathcal{D}}[\max_{\tilde{x}:\|\tilde{x}-x\|_p\leq\epsilon} |l(g(\tilde{x},W),y) - l(g(x,W),y)|], \\ &\leq L(W) + \mathbb{E}_{(x,y)\sim\mathcal{D}}[\max_{\tilde{x}:\|\tilde{x}-x\|_p\leq\epsilon} \lambda_p \|g(\tilde{x},W) - g(x,W)\|_p], \\ &\leq L(W) + \mathbb{E}_{(x,y)\sim\mathcal{D}}[\max_{\tilde{x}:\|\tilde{x}-x\|_p\leq\epsilon} \lambda_p \Lambda_p \|\tilde{x} - x\|_p], \\ &\leq L(W) + \lambda_p \Lambda_p \epsilon. \end{aligned}$$

This suggests that adversarial sensitivity can be controlled by the Lipschitz constant of the network. Another work (Xu & Mannor 2012) provided an upper bound for the generalization error $L(W)$ using the average loss on the training set, the Lipschitz constants λ_p, Λ_p, and the "covering number" of the input domain. This suggests that the Lipschitz constants also control the generalization. Hence, accuracy and robustness are not at odds here neither.

In an unclear explanation, the authors claim that the Lipschitz constant Λ_p of the model is more or less the product of the Lipschitz constant of each layer. Hence, if the Lipschitz constants of the layers are greater

than 1, Λ_p can grow exponentially with the depth of the network. Is that implying that deeper networks are more sensitive to adversarial attacks? I don't remember seeing anyone claiming that.

The Lipschitz constant of a layer k is basically the l_p matrix norm of the weights $W^{(k)}$. It is a little bit different for convolutional layers. Remember that $\|W^{(k)}\|_2$ is the *spectral norm* which is the largest singular value of $W^{(k)}$, and $\|W^{(k)}\|_\infty$ is the maximum 1-norm of the rows. When going through an aggregation layer, the Lipschitz constants are simply added. For activation function, it suffices to have Lipschitz constant less or equal to 1, which is the case for ReLU.

To control Λ_p, Parseval networks ensure that each layer has a Lipschitz constant less or equal to 1 (to avoid the exponential grow). Then, it uses an usual regularization scheme (like weight decay) to control the overall Lipschitz constant of the network. I personally don't see why nor how weight decay will control the Lipschitz constant of the network and there is no reference to explain this in the paper, which is curious since they tend to put a reference for virtually any affirmation.

Anyway, two modifications are used to ensure that the Lipschitz constant of each layer is smaller than 1:

1. Orthogonality of weight matrices. To control Λ_2 for linear and convolutional layer, it suffices to ensure that the largest singular value is 1. This can be achieve by ensuring that the rows of the weight matrices are orthogonal (or "Parseval tight frame" when the matrix is not square) because the singular values of an orthogonal matrix are all equal to 1. To control Λ_∞, it is possible to rescale each row to have 1-norm smaller than 1 (but this is not implemented in the paper).

2. Convex combination in aggregation layers. The coefficients of the combination are learnable. Since the combination is convex, Λ_p will remain smaller than 1.

Then, three computational tricks are proposed:

1. In order to stay on the manifold of orthogonal matrices (the *Stiefel manifold* if you want to flex), the authors propose to do one gradient step of the layer-wise regularizer:

$$R_\beta(W^{(k)}) = \frac{\beta}{2}\|W^{(k)}(W^{(k)})^T - I\|_2^2. \tag{1}$$

2. Select a subset of rows of $W^{(k)}$ uniformly and compute the gradient step on these rows only.

3. For the combination in aggregation layers to be convex, the coefficients must be projected onto the positive simplex after each gradient update.

To check the orthogonality of weight matrices, the authors simply compute the singular values and check that they are close to 1. As a side remark, it seems that weight decay leads to "sparse spectrum" (I guess it means lots of singular values close to zero), especially in the last layers, suggesting a "low-rank structure" (whatever that means). Then, the authors experimentally show that Parseval networks improve the robustness accuracy of vanilla models to adversarial examples. They claim that Parseval networks help for adversarial examples that are close to the original point, while adversarial training can help for further adversarial points (but of course there is no proof except some vague interpretations of experiments). It seems that Parseval networks converge faster than vanilla networks (because the weight matrices are "well conditioned" whatever that means) and that they use better their capacity. Let us detail this last point. The authors use a metric called **local covariance dimension**:

1. First, compute the activation's empirical covariance matrix $\frac{1}{n}\sum_{i=1}^n \phi_k(x_i)\phi_k(x_i)^T$ where $\phi_k(x)$ is the output of the k-th layer.

2. Then, compute and sort the eigenvalues of this covariance matrix $\sigma_1 \geq \ldots \geq \sigma_d$.

3. Finally, select the smallest integer p such that $\sum_{i=1}^p \sigma_i \geq 0.99 \sum_{i=1}^d \sigma_i$.

This gives the number of dimensions needed to explain 99% of the covariance. The same dimension can be computed for each class separatly (instead of all mixed together) by considering only the x_i belonging to a certain class. The results show that vanilla networks (even with adversarial training and weight decay) have high dimension in the first layer ($\sim 70\%$) and then very small dimensions in all other layers ($\sim 1\%$)

while the dimension remains high in all layers for Parseval networks. In this sense, Parseval networks "use the capacity better". This is intriguing because vanilla networks with adversarial training also increase the robustness but *not in the same way as Parseval networks*. The same phenomenon can be observed for each class. Moreover, for vanilla networks, the total dimension and the per-class dimension are the same (except for the first layer), while the per-class dimension of Parseval networks is smaller than its total dimension. The authors interpret this phenomenon by claiming that "Parseval networks contract the data of each class in a lower dimensional manifold (compared to the "intrinsic dimensionality of the whole data") hence "making classification easier"". Yes, there are double double quotes to show how much I find these kind of interpretations unconvincing. It sounds almost like a political speech : it seems clever, but in reality it is mostly empty. Why on Earth using the word "manifold" without doing any geometry or topology of any kind?

Other regularization methods Penalize the partial derivatives of each layer.

Label smoothing Label smoothing can only be used in classification tasks. Let m be the number of classes. Label smoothing consists in replacing the true distribution q (where $q_i = 1$ if i is the true label and $q_j = 0$ for $j \neq i$) by a "smoothed" distribution q^{LS} such that

$$q_i^{LS} = (1 - \alpha)q_i + \frac{\alpha}{m}.$$

It is a linear combination of the true distribution with the uniform distribution. In [15], Müller et al. show that label smoothing reduces the over-confidence of the model, but impairs knowledge distillation.

The largest eigenvalue suppression method introduced by Shen et al. [16] is equivalent to label smoothing.

In his unpublished work, Naddeo uses an orthogonal matrix to transform the dataset prior to training (orhtogonal pre-processing: OPP). The idea is to change the eigenvector associated to the largest eigenvalue, ideally to an orthogonal direction. Since the matrix is orthogonal, only the directions of the eigenvectors are changed, but not the eigenvalues. This is supposed to be an adversarial defense against black-box attacks. In black-box attacks, a attacker will select the direction of the highest eigenvalue in another model, expecting that this direction will be the same in other models trained on the same task (which is true because of transferability). But with OPP, this direction does no longer corresponds to the highest eigenvalue.

Then, Naddeo argues that his OPP method is the only one, along with largest eigenvalue suppression, to manipulate the "data manifold" using KL divergence (and more precisely the second-order approximation of KL divergence where the FIM appears). And between the two methods, OPP is the best (of course).

I am very much skeptical with OPP. It may be due to a misunderstanding from my side. Since the model has been trained on the transformed dataset, then, during inference, every input image must be transformed with the same transformation. In particular, an adversarial example will be transformed before going through the model. Hence, the adversarial direction will be transformed too, such that the transformed direction corresponds to the highest eigenvalue of the FIM! Then the question is: why does the method works experimentally? I don't know. Maybe, it is because the experimenters have "overfitted" the experiment (with hyperparameters, with multiple seeds etc.). Maybe, learning on a transformed dataset is harder, and the obtained model is less accurate but more robust for some reason. Maybe, it is due to the chosen metric (the fooling ratio): if the accuracy on the test set A_{test} is smaller but the accuracy on adversarial examples A_{adv} is constant, then the fooling ratio on the less accurate model will be *smaller* than on the more accurate one. The fooling ratio reduces to the function $F : x \mapsto 1 - k/x$ where $k = A_{avd}$ is a constant and $k \leq x \leq 1$. F is strictly increasing from $F(k) = 0$ to $F(1) = 1 - k$. Maybe this is expected (a less accurate model is expected to be less accurate on adversarial examples too, but if it is not the case, then the model is better).

2.2.2 Adversarial (re)training

The idea is to minimize

$$\frac{1}{N}\sum_{i=1}^{N} \max_{\delta \in \Delta(x_i)} \mathcal{L}(N(x_i + \delta), y_i),$$

instead of

$$\frac{1}{N}\sum_{i=1}^{N}\mathcal{L}(N(x_i), y_i),$$

where $\Delta(x_i)$ is a set of allowed adversarial attacks for x_i.

See Madry et al. [6] 2018, and Goodfellow et al. [4] 2015.

PGD training is efficient but slow. It is possible to accelerate the training by using computations from the backpropagation with respect the weights to accelerate the backpropagation with respect to the inputs. See YOPO (you only propagate once).

Adversarial training can also be improved by using another adversarial loss. See for example TRADES [17], or MART. Several improvements of adversarial training are mentioned in [18].

2.3 Certified defenses

See convex relaxation, Reluplex, and (trainable) certificates. A certificate is a lower bound for the minimal perturbation distance. There are lots of works on certificates.

See Zhang et al. [17] for a theoretical upper bound for the trade-off between robustness and accuracy.

2.3.1 Randomized smoothing

As far as I know, randomized smoothing is the only certified defense that is scalable to large models. The idea is very simple. Suppose you have a trained model $F : \mathbb{R}^n \to \mathcal{Y}$ for a classification task. Let $x \in \mathbb{R}^n$. Choose a parameter $\sigma > 0$ and consider the probability space $(\mathbb{R}^n, \mathcal{B}(\mathbb{R}^n), \mathbb{P}_x)$ where the distribution of $\mathbb{P}_x$ is $N(x, \sigma^2 I)$. Now, we can see F as a random variable.

We can define a new classifier $G : \mathbb{R}^n \to \mathcal{Y}$ (that is *not* a neural network) by:

$$G(x) = \arg\max_c \mathbb{P}_x(F^{-1}(c)),$$

It can be shown (by Cohen et al. [19]) that, if there is a class $c_A \in \mathcal{Y}$ and two probabilities $\underline{p_A}, \overline{p_B}$ such that:

$$\mathbb{P}_x(F^{-1}(c_A)) \geq \underline{p_A} \geq \overline{p_B} \geq \max_{c \neq c_A} \mathbb{P}_x(F^{-1}(c)), \tag{2}$$

then we have $G(x + \delta) = c_A$ for all $\|\delta\|_2 \leq R$ where:

$$R = \frac{\sigma}{2}\left(\Phi^{-1}(\underline{p_A}) - \Phi^{-1}(\overline{p_B})\right), \tag{3}$$

with Φ^{-1} the quantile function of the standard normal distribution.

Moreover, Cohen et al. [19] showed that this bound is tight, i.e., if $\underline{p_A} + \overline{p_B} \leq 1$, then for any perturbation δ such that $\|\delta\|_2 \geq R$, there exists a classifier F verifying equation 2 for which $G(x + \delta) \neq c_A$. Hence, the set of perturbations to which the smoothed classifier is provably robust is precisely an l_2 ball.

Thus, smoothing with other distributions (other than Gaussian) may lead to certified guarantees for other l_p norm, or even other type of attacks. Hao et al. may have done this for spatial attacks [20].

The radius depends linearly on σ. But most importantly, the radius is larger when $\underline{p_A}$ is close to 1 and $\overline{p_B}$ is close to 0. The higher is σ, the less likely this will be true, thus there is a trade-off. This trade-off depends on the ability of the model F to be robust to Gaussian noise. Hence, this result shows a direct link between robustness to Gaussian noise and certified robustness to adversarial attacks.

In practice, it is impossible to compute $\mathbb{P}_x(F^{-1}(c))$. Cohen et al. There are two problems:

1. Prediction → estimate what class is predicted by G.
2. Certification → estimate the radius R

For prediction, the authors of [19] propose a method to estimate $\mathbb{P}_x(F^{-1}(c))$ with Monte Carlo sampling. The method guarantees that the predicted class for $G(x)$ is not incorrect with probability $1 - \alpha$ where α is a chosen risk. If the model is unsure, it will abstain to make any prediction. For a fixed α, in order to avoid abstaining, we must sample a large number of points. The prediction methods is the following (I don't if these algorithms are for two classes or any number of classes, I guess they are for two classes):

1. Sample N points from $f(\mathcal{N}(x, \sigma^2 I))$.
2. Let N_A and N_B be the number of points in N associated with largest and second largest classes.
3. For c_A to be the predicted class of G, the probability of obtained c_A from the $N_A + N_B$ points of the first two classes must be greater than $1/2$. So we compute the probability (p-value) that there are N_A successes among $N_A + N_B$ trials of a Binomial distribution with parameter $1/2$. If this p-value is smaller than α, return c_A, else, abstain.

The certification is a little bit more complicated. To compute equation 3, we need to evaluate a lower bound $\underline{p_A}$. Then, we set $\overline{p_B} = 1 - \underline{p_A}$.

1. Sample a small number N_0 of points from $f(\mathcal{N}(x, \sigma^2 I))$.
2. Take a guess for c_A as the largest class among the N_0 points.
3. Sample a larger number N of points from $f(\mathcal{N}(x, \sigma^4 I))$ (I don't know if there is an error here in [19] and it should be σ^2 instead of σ^4). Let N_A be the number of c_A among the N points.
4. Compute the confidence interval $[p, +\infty)$ of the Binomial distribution with N trials, N_A successes, and risk α. We set $\underline{p_A} = p$.
5. If $\underline{p_A} > 1/2$, return the prediction c_A and the radius $\sigma \Phi^{-1}(\underline{p_A})$, else, abstain.

How can we extend randomized smoothing to regression (maybe it has already been done)? Can we interpret randomized smoothing with information geometry?

2.4 Adversarial detection

2.4.1 Auxiliary models to classify adversarial examples

2.4.2 Using statistics

PCA, statistical test to see if two examples are from the same distribution.

2.4.3 Checking prediction consistency

If we modify a adversarial example, the output will likely change, while it will not change for a benign example.

2.4.4 Extended Variational Density Propagation (exVDP)

In [21], Dera et al. introduce an extended variational density propagation (exVDP) framework for propagating uncertainty in Convolutional Neural Networks (CNN). The weights of the filters of the convolutional layers are modeled as Tensor Normal Distribution. The weights of the fully-connected layers are also modeled as Gaussian vectors. As part of the VI framework, the ELBO is used as the objective function of the network. To estimate the expected log-likelihood (first term of the ELBO), the first and second moments of the weights distributions are propagated through the various layers (convolutional, activation, pooling, fully-connected). Once the posterior distribution is known, the expected log-likelihood is estimated with Monte Carlo sampling, and the full ELBO can be derived for the CNN. The exVDP is compared to other Bayesian networks as well as to classical CNN, on MNIST and CFAR-10 datasets. exVDP is shown to be able to resist Gaussian noise and adversarial attacks far better than other frameworks. Moreover, the propagated variance can be used to quantify the uncertainty of the network.

In [22], Dera et al. apply the exVDP framework to the classification of synthetic aperture radar (SAR) images. These are satellite images than can be used to classify the type of surface in each part of the image (water, crops, buildings etc.). Once again, the exVDP framework is able to resist Gaussian noise and adversarial attack. It is also possible to generate an uncertainty map allowing to visualize the uncertainty of the prediction of the network on each area of the image.

3 Proposed Explanations

From Xu et al. [3] §5.2: many recent works hypothesize that it might be impossible to build optimally robust classifier. For example, one study claims that adversarial examples are inevitable because the distribution of data in each class is not well-concentrated, which leaves room for adversarial examples. In this vein, another work claims that to improve the robustness of a trained model, it is necessary to collect more data. Moreover, the authors in another work suggest, even if we can build models with high robustness, it must take cost of some accuracy.

From Akhtar et al. [23] §IX.C.: Bubeck et al. argued that adversarial vulnerability of classifiers in high dimension is "likely not due to information theoretic limitations, but rather it could be due to computational constraints". They provided evidence to support the hypothesis that "identifying a robust classifier from limited training data is information theoretically possible but computationally impossible". Interestingly, their evidence weakens the notion that identifying a robust classifier requires huge amount of training data.

3.1 High dimensionality

3.1.1 Concentration of measure in high-dimensions

Was developed by Gilmer et al. [10] (2018), Mahloujifar et al., [24] (2018), Fawzi et al. [25] (2018), and Shafahi et al. [26] (2019).

The high dimensionality of the input space can present fundamental barriers on classifier robustness. Theoretical and experimental proofs have shown that, *for certain data distributions*, any decision boundary is close to a large fraction of inputs and hence no classifier can be robust against small perturbations. This theory want to explain why, despite lots of proposition for adversarial defenses, there are still new attacks that manage to break thoses defenses.

For now, I haven't engage with the literature on this topic, but it seems to be linked with the curse of dimensionality. As far as I understand, an intuition behind this phenomenon is the relation between the unit cube and the unit sphere. Imagine that the decision boundaries are the faces of the unit cube. We pick a point at random in the unit cube, with an uniform distribution. It can be shown that the higher the dimension, the closer will this point be to the decision boundaries (but I don't know the precise relation). To see that, consider the volume of the unit cube which is not contain in the unit sphere. When the dimension increases, this volume occupies a larger portion of the unit cube, while the unit sphere occupies a negligible volume. So, "most" of the points of the unit cube are outside the unit sphere, i.e., there distance to the origin is larger than 1.

This theory certainly explains partially the ubiquity of adversarial attacks. It provides bounds on the robustness that can be achieved by any classifier of certain datasets. It is not incompatible with other explanations. However, I think this theory has several drawbacks:

- It implies that adversarial attacks are far less frequent for small dimensions. It may be true but I am not sure. I can imagine models in two dimensions that have overfitted the dataset so much that there are no longer robust at all (but this might not be representative of the "typical behavior" in two dimensions).
- How is it possible that some models have been trained to be robust, and are actually fairly robust?
- How is it possible that humans are not sensitive to adversarial attacks if they are only due to dimensionality issues?
- I don't know the exact content of the theory, but it seems that it has to make assumptions on the distribution of the dataset. These assumptions may not be representative of real-life datasets.

Update after reading Gilmer at al. [10] The theory behind [10] is actually quite simple. They use a synthetic dataset for a binary classification task (two concentric spheres). First, they show experimentally that a model with a very low error rate still has adversarial attacks *on the data manifold* (i.e., on the same

sphere as the original point). Then, they provide an upper bound for the adversarial robustness which depends on the error rate and on the dimension of the problem:

$$d(E) < O\left(\frac{\phi^{-1}(1-\mu(E))}{\sqrt{n}}\right),$$

where $\mu(E) = \mathbb{P}_{x\sim p}(x \in E)$ is the measure of the error set, $d(E) = \mathbb{E}_{x\sim p}[d(x, E)]$ is the average distance between a point sampled according to the dataset distribution and the error set, ϕ^{-1} is the quantile function of the standard normal distribution, and n is the dimension of the problem. This is a consequence of the concentration of measure in high dimensions. In high dimension, almost all the points of the sphere are close to the equator *for any equator*[1]. Hence, even if E has small measure and is well concentrated at one area of the sphere (and not dispersed accross the sphere), then E will be close to every equator. In particular, if there is an equator such that all of E is on one hemisphere, then the opposite pole (which is the farthest point to E) will be at a distance $\approx \sqrt{2}R$ from E where R is the radius of the sphere[2]. It must be emphasized that this result is highly dependent on the geometry of the synthetic dataset. As far as I know, there isn't similar bounds for specific real-world datasets like MNIST or CIFAR-10, and even less for arbitrary datasets. Thus, I am not very much convinced by this result, but I think there are still important lessons to learn from this paper, and particularly from the appendix.

- There may be an upper bound on the adversarial robustness (given an error rate and the dimension of the problem), but this bound seems to be high for real-world datasets and thus is not the main reason behind adversarial vulnerability of current deep learning models. For example, Appendix G shows that this bound is not reached on MNIST, even with adversarial training.

- Accuracy and robustness may *not* be antagonist since the upper bound on the robustness is proportional to the accuracy of the model (more precisely to the generalization capability of the model).

- Models trained with cross entropy and SGD seem to achieve low error rate *without really understanding the task*. The models learn to detect statistical correlation by "adding wrong numbers that add to good predictions". Even a model initialized with zero error rate and zero adversarial vulnerability (but nonzero cross entropy) will converge to a model with non zero (but very small) error rate and adversarial vulnerability. This is because the cross entropy is not perfectly matched with accuracy, and the model will reduce the average cross entropy *while dramatically increasing the worst-case cross entropy for a very small number of points*. It seems that **something is very wrong about the way models are trained**.

- Models are even able to achieve very low error rate *while using only a fraction of the entire input dimensions*, while it is impossible to achieve zero error rate without using all dimensions. Hence, understanding a (simple) task and reaching zero error rate is *very* different from achieve a very low (but nonzero) error rate by looking at statistical correlations.

- This paper still relies on the "manifold hypothesis" by assuming that the data lie on low dimensional manifolds (the spheres). However, the paper rejects the explanation of adversarial vulnerability claiming that the adversarial examples are off-manifold (i.e., the "boundary tilting" hypothesis).

- There is nothing special about adversarial examples in the sense that, for a very accurate model, any incorrectly classified point is close to a correctly classified point. In other words, every incorrectly classified point *is* an adversarial example! Hence, adversarial detection is doomed to fail!

- As it is very common, the paper assumes that the dataset stems from a (really-existing but unknown) probability distribution $p(x)$. I don't know why, but this assumption makes me feel awkward.

[1] Intuition. Choose any orthonormal basis. Let $x = (x_1, \ldots, x_n)$ be a random point uniformly sampled over the unit sphere. An equator is the set of points on the sphere such that $x_i = 0$ for some i. Each x_i has the same distribution and we have $\sum x_i^2 = 1$. Since n is large, each x_i is very likely to be small. Hence, the point x is very likely to be close to all equators.

[2] The maximum distance between two points of the sphere is $2R$.

3.1.2 Linear explanation

This was the first explanation to achieve a consensus before being challenged. It was proposed by Goodfellow et al. [4] in 2015.

In [2], it was supposed that adversarial examples are due to extreme nonlinearity of deep neural networks, combine with insufficient regularization (overfitting). The idea was to imagine that adversarial examples are a dense set with measure zero, such that all examples from the training and test sets are *not* adversarial.

For Goofellow et al., these hypothesis are unnecessary (but adversarial examples existing for linear model doesn't mean that nonlinearity does not worsen the problem, isn't it?). Linear behavior in high-dimensional spaces is sufficient to cause adversarial examples. Using this assumption, they propose the FGSM attack which proves to be very effective. They show that adversarial training provides additional regularization and reduces the model's vulnerability, while generic regularization methods (dropout, pretraining, model averaging) do not reduce the vulnerability to adversarial examples. They claim that using very nonlinear models increases the robustness, but this claim has been challenged (see [13]).

For Goodfellow et al., the trade-off is between **models that are easy to train due to their linearity and models that use nonlinear effects to resist adversarial perturbation.** For them, the solution to adversarial attacks lies in the efficiency of the optimization methods. They claim that linear models cannot escape adversarial examples ([27] claim that this is not true), so only nonlinear models (with hidden layers) can be trained to resist adversarial attacks.

The reasoning is as follows. We consider a linear model $w^T x$ for some weight vector w. Let $\tilde{x} = x + \eta$ be an adversarial example. We have $w^T \tilde{x} = w^T x + w^T \eta$. Assume that we impose a bound on the max norm of the perturbation: $\|\eta\|_\infty < \epsilon$. Assume that the average magnitude of the weight w is $m = \sum_i w^i / n$. We can maximize the increase caused by $w^T \eta$ by choosing $\eta = \epsilon \text{sign}(w)$. In this case, we have $w^T \eta = \sum_i \epsilon |w^i| = \epsilon mn$. For a fixed-size perturbation ϵ, the increase in the activation grows linearly with n. To conclude, the problem is due to:

- The model being linear. More precisely, a linear model is forced to attend exclusively to the signal that **aligns most closely with its weights**, even if other signals have greater amplitude (but are not aligned with the weights).
- The dimension of the input being high.

Adversarial examples generalize across different models because different models learn similar functions when trained on the same task. They also claim that the **direction of perturbation is more import than the specific point in space**, which is a counter-argument for adversarial examples begin "dense" (see Universal Adversarial Perturbations). This also explains why adversarial perturbations generalize across different clean examples. They also show that ensemble methods are not resistant to adversarial examples.

According to [27], the linearity argument is unconvincing because, even if $w^T \eta$ grows linearly with n, this is also the case for $w^T x$, such that **their ratio stays constant.**

3.1.3 Piecewise linear decision boundaries

Proposed by Shamir et al. [28] in 2019.

They study $\mathbb{R}^n$ with the L_0 (Hamming) metric and explain why a deep neural networks trained to distinguish between m classes can be fooled with adversarial examples of Hamming distance $\sim m$.

They work on targeted attack (which is harder than untargeted attack). At first, it seems mysterious that all the classes defined by neural networks in the input space $\mathbb{R}^n$ are intertwined in a *fractal-like way* so that *any point in any class is simultaneously close to all the boundaries with all the other classes.*

The problem with this work (as mentioned by [13]) is that they are using the L_0 "norm" (number of elements that changed) and their arguments cannot extend to other norms. They show that we can choose a small number of directions ($\sim m$), then move along these directions *as far as we want* and reach any other class. This is particularly true if the input dimension is high. Note that this seems far less mysterious when stated like this. In particular, this type of attack does not respect the input range (for example, image pixels may be bounded between 0 and 255). Finally, it seems that this paper doesn't tell us anything useful about adversarial attacks in the real world.

3.2 Overfitting

3.2.1 Insufficient data

Proposed by Schmidt et al. [29] in 2018.

Adversarial examples arise due to insufficient information about the true data distribution. Robust models may achieve robustness by reducing the effective/useful amount of information in the training data (discarding non-robust features).

3.2.2 Boundary tilting

Proposed by Tanay and Griffin [27] in 2016.

The decision boundaries extend beyond the submanifold of the dataset (i.e., the data leaf) and can be lying close to it under certain circumstances. Regularization is the solution to adversarial examples by preventing finite-sample overfitting. This explanation is in contradiction with the distillation hypothesis which suggests that adversarial examples are made of features inherent to the data distribution, hence lying on the submanifold of the dataset.

After skimming through the paper, its seems that their mathematical analysis focuses on linear classifiers, like SVM or logistic regression. They use in-depth geometrical analysis but using very classical tools of Euclidean geometry. They don't seem to explain how their framework extends to deep neural networks. One way to see it is to consider that their analysis investigates the link between the penultimate layer and the output (since SVM can use non-linear kernels).

Anyway, I mostly agree with their proposition. I think that my proposition will mostly extend their by using tools from Riemannian geometry. The idea is to use a metric that is meaningful for the network by linking the input with the output, instead of using norms that are only related to the input space and that are somewhat arbitrary.

3.2.3 Noise

Several papers (I don't list them for now but they can be found in [13]) showed that robustness to adversarial attacks is directly linked to robustness to random noise. Moreover, other papers use random noise during training to construct robust classifiers.

This is not an explanation of adversarial examples per se, but it indicates that our theory must explain adversarial examples and random noise with similar mechanisms.

3.3 Non-robust features

This explanation was introduced by Ilyas et al. [13] and Tsipras et al. [30] in 2019. It can be seen as a form of dataset distillation since their goal is to show that we can achieve robustness **by modifying only the training set and not the training process.** It shows that adversarial vulnerability is mainly a property of the dataset.

Let $D = \{x\}$ be a dataset. We want to create a new dataset of the same size $D_r = \{x_r\}$ using a map $x \mapsto x_r$. Consider a model N_r robustly trained (using the framework of the previous subsection). Consider the penultimate layer of N_r denoted $g(x)$, i.e., the output of the network is $\hat{y} = \text{softmax}(W.g(x)+b)$. Sample an input x_0 uniformly from D. Then, consider $f(x_k) = \|g(x_k) - g(x)\|$. To be continued ...

This theory provides an explanation of **adversarial transferability**. This is a natural consequence of the existence of non-robust features. Other works (see [13]) have studied adversarial transferability from a theoretical perspective, using *simple models* or *unbounded perturbations*. In particular, Tramer et al. [31] (2017) showed empirically the existence of **adversarial subspaces** i.e., linear subspaces containing only adversarial perturbations. Moreover, they showed that there is significant overlap in the adversarial subspaces between different models, thus explaining transferability. The directions of these subspaces might correspond to non-robust features.

The non-robust features theory also explains **universal adversarial perturbations** i.e., perturbations that work on different inputs. These perturbations correspond to non-robust features that appear in different images/input points.

In [13], other works are mentioned to study the manipulation of dataset features and its impact to robustness. It seems that removing information from the training set improve the robustness, which is once again coherent with the non-robust features theory.

There is a trade-off between accuracy and robustness. If we refuse to use non-robust features, we lose accuracy but gain robustness. However, discarding non-robust features means that the model is more interpretable.

An answer to [13] by Nakkiran shows that it is possible to construct an attack which is non-transferable by design. Hence, this type of attack is really a "bug" of the model under attack and not a "feature" of the dataset. Nakkiran speculates that Ilyas et al. were unable to detect the bug-type attack because they use the PGD method to robustify their dataset. However, the PGD method is intrinsically prone to create attacks "along directions of the data" that are transferable, and hence that are of the feature-type. The adversarial examples you obtain depend on the attack you use!

3.3.1 Other theoretical constructions

As mentioned in [13], there is a paper that explain adversarial examples with *computational constraints* and another with *model complexity*. Both papers construct datasets that admit perfectly accurate yet non-robust classifiers. To do so, they implicitly use the concept of *non-robust features* by adding a low-magnitude signal to the input that encodes the true label. This low-magnitude signal is a non-robust feature, since it allows a classifier to achieve perfect accuracy but is highly sensitive to small perturbations.

3.4 Information geometry

In this subsection, I review most of the literature applying "information geometry" to the problem of adversarial robustness in machine learning.

In 2016, Miyato et al. [32] introduce a regularization called virtual adversarial training (VAT) based on Kullback˙Leibler divergence. The goal is to locally smooth the output distribution in order to improve generalization. Let x be a training point. Let $p(y|x,\theta)$ be the distribution over the output defined by a model parameterized by the weights θ. Define:

$$r_{vat} := \arg\max_r \{\mathrm{KL}[p(y|x,\theta)||p(y|x+r,\theta)]; \|r\|_2 \leq \epsilon\}.$$

Then, the regularization term at x is $-\mathrm{KL}[p(y|x,\theta)||p(y|x+r_{vat},\theta)]$. Contrary to L_p regularization, it is invariant to reparametrization and act locally instead of globally. Contrary to adversarial training, it does not require the true label. When $p(y|x,\theta)$ is not a well known family of distributions, the authors propose to use the second-order Taylor approximation of $\mathrm{KL}[p(y|x,\theta)||p(y|x+r,\theta)]$ around $r=0$. Thus, r_{adv} can be approximated by the first dominant eigenvector of the Hessian matrix. Even if never mentioned in the paper, it turns out that this Hessian matrix is the Fisher information matrix.

In 2017, Nayebi & Ganguli [33] use a regularization method aiming at saturating the network, i.e., encouraging activations to be in the saturating regime of the nonlinearity where the derivative is close to zero. It is similar to Jacobian regularization which also push towards smaller derivatives but less computationally expansive. The rational is based on the explanation provided by Goodfellow et al. [4] in 2015, claiming that adversarial vulnerability is due to the models being too linear. The l-th layer is $x^l = \phi(h^l) = \phi(W^l x^{l-1} + b^l)$. The regularization term takes the form:

$$\lambda \sum_{l,i} \phi_c(h_i^l)$$

where the sum is taken over all layers l and all components i in the layer l. The function ϕ_c is called the complementary function. It returns the distance to the closest point where the derivative of ϕ is zero.

Note that they are using what they call "*annealing*" for λ, which consists in increasing λ during training. Moreover, some of their values for λ are as small as 10^{-8}. This is exactly what I did for the isometry regularization.

The authors use information geometry to analyze their method. They define the Fisher information metric (FIM) G^F on the output space seen as a space of distributions, then introduce the pullback metric on the input space $G^{in} = J^T G^F J$. They remark that the Jacobian J is of geometric interest by itself,

since the number of large singular values of J determine how many directions in input space the network's input-output map is locally sensitive to. The main finding is that both the metric length element and the Jacobian largest singular values rise quickly when a class transition occurs then decrease. The transition is even sharper for satured networks than for vanilla networks. The authors claim that it is a good property for adversarial robustness since it means that the network is locally flat, such that gradient-based adversaries have a hard time finding attack directions.

Notice how their findings are very similar to mine. However, their interpretation is very different. They claim that having sharp class transitions can obfuscate the adversary. It may be true for gradient-based adversary, but I am not sure. Maybe the direction is still the right direction to attack even if the singular value is small. And I read that gradient-obfuscating methods are very weak. They do not really improve robustness to adversarial attacks, they just provide a defense for a very specific type of attacks.

My interpretation is this: we don't understand what's going on. Nothing happens for a long time, and suddenly, there is a spike, the class changes and then nothing happens again. Why is there a spike here and not there? How can we control them? We don't know. Our defenses are only local (i.e., based on the derivatives at the training points), so the model can satisfy us by providing a "flat" or "saturated" landscape around the training points, and then do whatever it wants between the training points. We must control the training "globally". This is linked to generalization. We must use our knowledge of the architecture of the models (this is not a black box!).

Moreover, one might say that "geometry" consists in finding global properties using locally defined objects. Thus, despite what is often claimed in these papers, studying the spectrum of the FIM is not geometry.

In 2019, Zhao et al. [11] introduce the one-step spectral attack which consists in using the first eigenvector of the Fisher information matrix as the direction of attack. Moreover, the eigenvalues of the Fisher information matrix can be used as features to train a model to detect adversarial examples (which I find very dubious).

In 2019, Martin and Elster [34] investigate further how the Fisher information matrix can be used to detect adversarial examples. However, the authors do not use the Fisher information matrix with respect to the inputs as in [11] but with respect to the network's weights.

"*We should point out once more that whenever we speak of curvature this should be understood in loose sense and* not *in the one of differential geometry. Computing the actual, say Ricci-, curvature for the statistical manifold of a neural network is probably unfeasible.*"

Tron et al. [12] in 2022.

Grementieri and Fioresi [35] in 2021.

4 Conjectures

4.1 Robustness is linked to the extrinsic curvature of decision boundaries [Week 29]

- The decision boundaries are submanifolds of dimension $n-1$. The kernel leaves that intersect a decision boundary are entirely contained in it. The kernel leaves induce a foliation on each decision boundary.
- To obtain the optimally robust network, the solution might be to "flatten" the decision boundaries, which can be achieved by flattening the kernel leaves. "Flattening" the kernel leaves mean reducing their extrinsic curvature as submanifolds of $\mathbb{R}^n$. However, the kernel leaves are not Riemannian manifolds (because G_x is 0 on them), so I don't know how this impacts the notion of extrinsic curvature.
- The image foliation and the kernel foliation are transverse to each other, so there must exist a link between their curvature. Since the dataset has to belong to the data leaf, the geometry of the dataset directly impacts how much the kernel leaves can be flattened.

4.2 Geometrical interpretation of supervised learning [Week 29]

Supervised learning is traditionally described using statistical and optimization frameworks. Similar to the geometrical interpretation of margins in Support Vector Machine, it might be possible to reformulate

supervised learning with a geometrical framework.

What is learning a supervised learning task? It consists in finding a Riemannian submanifold such that the expected Fisher-Rao distance between points of the same class (or points with similar output for regression) is minimized. This Riemannian submanifold is the data leaf and the Fisher-Rao distance is calculated along the data leaf. The Fisher metric is not defined outside this submanifold.

The Fisher-Rao distance have nothing to do with the Euclidean distance. Using normal coordinates may help for calculation/visualization.

Where are the adversarial points (for classical adversarial attack methods)? There are two possible answers.

1. They are on the data leaf. They aim in the direction where the Fisher-Rao distance increase the most for a given Euclidean distance (or any other l_p norm). In a classification setting, this consists in looking for the closest decision boundary along "normal data" (no noise). This seems to be the case for the OSSA attack from Zhao et al. 2019. The eigenvectors of the Fisher matrix associated to nonzero eigenvalues are in the data leaf (at least for a small Euclidean displacement). The efficiency of this attack might rely on the concentration of measure (i.e., there always exists a decision boundary close to any point.).

2. They are not on the data leaf. This is the assumption of Grementieri & Fioresi where they indicate that the FGSM attack is not likely to be in the data leaf. Maybe, an adversarial attack is a "good" combination of moving towards a decision boundary in the data leaf, and adding noise? Or maybe (this is what I believe), the decision boundaries tend to curve when they exit the data leaf such that they stay close to the data leaf for every point of the data leaf. An adversarial attack will have to move cleverly to reach the decision boundary as fast as possible. Since it depends on the curvature of the image/kernel foliations, the initial vector of the attack needs to combine an image direction with a noise direction.

Here, I list the questions raised by this theory.

- We need to experimentally (and even theoretically) see where the classical attacks moves in order to know if they stay in the data leaf, and if no, how far they are going, and how they interact with the image foliation.

- If the second hypothesis is true, then the robustness is directly linked to the curvature of the foliations. A natural question is then: for a given data leaf (that maximize the accuracy), is it possible to choose/enforce a foliation with minimal curvature?

- The second hypothesis might explained the OSSA. Maybe, when the eigenvalue is large, the curvature is maximum, and this direction maximizes the KL divergence *while leaving the data leaf and reaching the closest decision boundary*. This may explain why, according to Zhao et al., the amplitude of the eigenvalues can be used to detect adversarial examples: high eigenvalues correspond to high curvature (at least locally) of the data leaf. The question is: is there truly a link between Riemannian curvature and eigenvalues of the metric? If yes, can this link explain OSSA?

- I still have to connect these conjectures with the distillation interpretation. What are "non-robust features" in my framework? How can we avoid relying on them? According to the Ilyas et al. [13], adversarial examples consist of features inherent to the data distribution, since they can encode generalizing information. Hence, these adversarial examples seems to lie on the data leaf.

- How can I explain overfitting with the geometrical interpretation.

- How can I connect my conjecture with the linear explanation. It seems that more curvature means less linearity, so the two explanations are contradictory.

- What about the concentration of measure explanation? Is it orthogonal/complementary to my explanation? Or maybe, both explanations are saying the same thing, albeit differently. My theory might be a continuation of the boundary tilting explanation, with Riemannian geometry notions.

- In order to be useful, my theory must work on general deep learning models (and not only on simple models), and must take into account the range of the inputs (maybe by working on manifold with boundary). It must explain generalization since "certified" training does not preclude adversarial examples from the test set.

4.3 Adversarial vulnerability is entirely characterized by the data leaf [Week 30]

4.3.1 Adversarial cylinders, certainty/uncertainty volumes, anti-adversarial examples

Can we identify submanifolds of adversarial examples? Yes. Given any point p in the data leaf that is an adversarial example, **any point in the kernel leaf of p is an adversarial example**. It is very likely that there is a neighborhood of p where all the points are adversarial examples. Then we could define a **adversarial cylinder** which is the extension of this neighborhood along the kernel foliation.

Indeed, any adversarial example is in one of these adversarial cylinders. In other words, the adversarial examples are entirely characterized by the the adversarial examples contained in the data leaf, that I call *adversarial data examples*. How can we characterize these adversarial data examples? An adversarial data example must be:

- Close to some original non-adversarial example according to some data space relevant (or human relevant) distance. "Close" is a vague term since it depends on the l_p norm you choose, which can make significant difference in the notion of "closeness". Then, it also depends on the chosen budget, which is completely arbitrary. It may be possible to define this "closeness" more objectively by saying that the adversarial example should be closer to any non-adversarial example of the original class than to any non-adversarial example of the other classes. "Non-adversarial example" is even more vague (this is entirely dependent on the underlying distribution of the data, which is unknown)! At least, we can consider the training and test points to be non-adversarial example (if the datasets were not augmented with adversarial examples of course).

- Far from the original non-adversarial example according to the Fisher-Rao distance. The exact definition is that the adversarial example should have a difference predicted class (i.e., a different argmax of the distribution) than the original. But this criteria is discrete, hence non-differentiable. But if the model is confident both on the original and the adversarial example, then the Fisher-Rao distance between the two must be high. This motivates us to define **certainty volumes** and **uncertainty volumes** where the entropy of the distribution is higher (once again, the threshold between the two is arbitrary). Original and adversarial examples must belong to certainty volumes. In uncertainty volumes, the model should answer "I don't know". What we want is to change the dataset/training process such that the uncertainty volumes encompass all adversarial examples (as well as noisy examples?).

Conjecture: any continuous curve from a non-adversarial point to an adversarial example crosses a uncertainty volume. Hence, certainty volumes containing a non-adversarial point cannot contain adversarial examples.

If the entropy of the distribution at a point p is high, this point p is an **anti-adversarial example**, i.e., the model is unsure how to classify p despite p being clearly recognizable by humans. Instead of extending the uncertainty volumes to adversarial examples, we should rather extend the certainty volumes to anti-adversarial examples! This won't impact the accuracy of the model (but maybe, the training would be longer?). However, this could harm the generalization capability of the model since this could destroy "useful" certainty volumes containing non-adversarial points (for example points from the test set).

4.3.2 Regularizing the data leaf

The fundamental issue with adversarial examples is the discrepancy between the "data distance" or "human distance" (norms l_2, l_∞, l_0) and the Fisher-Rao distance (along the data leaf) / the extrinsic curvature of the leaves (across leaves).

Conjecture: one can reduce the adversarial vulnerability by regularizing both the Fisher-Rao distance and the extrinsic curvature.

- The local data matrix should be as close as possible to an homothety matrix i.e., the condition number of the local data matrix should be close to 1 $\rightarrow$ **already done by Shen et al. 2019.**
- The extrinsic curvature should be as close as possible to 0.

This might be achievable by adding regularization terms to the loss function. However, this is will certainly harm the accuracy and slow the training (since it might be costly to evaluate the condition number and the extrinsic curvature). This solution is very boring because it is yet another regularization method, and I want to believe that it is possible to achieve robustness without harming the accuracy, and without slowing *too much* the training.

4.3.3 Optimal targeted attacks along the data leaf are geodesics along eigendirections

Conjecture: Each eigendirection of the local data matrix corresponds to a targeted attack direction. Indeed, since $\dim G(x, \omega) = C - 1$, then each direction might correspond to reaching a specific class. More precisely, the shortest path along the data leaf (according to the Fisher-Rao distance) between the starting point and the closest different class might be a the curve whose velocity at each point is a specific eigenvector of the local data matrix. This can be assessed experimentally.

4.4 Geometrical study of the pullback metric across layers [Week 30]

We want to study the kernel foliation with respect to the input of each layer and see how it evolves when we go from one layer to the next. When and how do adversarial examples appear (some works say that they appear at the first and last layers)? If we are to fully understand the phenomenon of adversarial vulnerability, we must precisely describe how the local data matrix evolves when "backpropagating" from one layer to the preceding one. We must focus on the specificity of each type of layer, while preserving a certain generality in order to easily extend our results to other family of layers. Here, we will focus on classification task using a simple CNN model (LeNet-like) with four type of layers:

- Convolutional layer with ReLU activation
- Max pooling layer
- Fully connected layer with ReLU activation
- Fully connected layer with Softmax activation

The attack budget is measured with the l_2 norm. This is the easiest norm for geometry since it is associated to the Euclidean metric. We will deal with other l_p norms later.

First, we need to describe the FIM in the output space. Let m be the number of classes. The output space is a $m-1$-dimensional manifold partitioned into m connected open sets (one for each class) called **decision manifolds**, separated by closed submanifolds of dimension $m-2$ (where two classes have the same probability) called **decision boundaries**.

When we go back through the previous layer, we focus on the data leaf (the other leaves are "copies" of the data leaf) in the **hidden space** i.e., the space of features between two layers. So, restricting the layer map to the data leaf, what is the pre-image of each decision manifold in the data leaf? When going back through the entire network, we conjecture that each decision manifold is scattered in the entire data leaf with multiple connected components. We must understand this phenomenon. The only constraint is that almost all training points are in the right decision manifold (the right label/class), and *far from any decision boundaries in the Fisher-Rao distance.*

Adversarial vulnerability may be due (or at least worsen) by the data leaf having high extrinsic curvature when endowed with the induced Euclidean metric. But, we assume for now that the extrinsic curvature is low and that the data leaf is almost flat in the Eulidean metric. This must be proved later. Here, we focus on explaining adversarial vulnerability in the data leaf. Adversarial vulnerability is explained by having each training point surrounded by connected components of all decision manifolds. These adversarial connected components (that we call **adversarial volumes**) are close to the original point in the Euclidean metric but far from it in the Fisher-Rao metric. Moreover, it seems that these adversarial volumes *do not contain any*

training point. We must understand how they appear. An interesting question is whether this concentration of adversarial volumes from all classes arises only close to training point, or if the entire data leaf has this structure.

Let us summarize our assumptions:

1. Classification task
2. The model is a ReLU Convolutional Neural Network, it is smooth (we neglect the ReLU non-differentiable point)
3. l_2 norm for the attack budget
4. Each hidden space has a dimension strictly greater than m
5. The local data matrix has constant rank in every hidden space so that the data leaf exists (we know that the distribution $x \mapsto (\ker G(x,\omega))^\perp$ is involutive)
6. The model is perfectly trained i.e., every training point is correctly classified
7. The data leaf has zero extrinsic curvature in the Euclidean metric

4.4.1 Geometry of the probability simplex [Week 32]

Amari 1985, p24 and p31.

Using the softmax function, the last layer outputs $p_1(x), \dots, p_m(x)$. Since we have $\sum_{i=1}^m p_i(x) = 1$, we can use $p_1 = p_1(x), \dots, p_{m-1} = p_{m-1}(x)$ as a coordinate system on the output space. The output space is a $m-1$-dimensional manifold called the probability simplex. It can be seen as the embedded submanifold of $\mathbb{R}^m$ defined by $\sum_{i=1}^m p_i = 1, p_i > 0$.

The probability mass function (i.e., probability density function with respect to the counting measure) is

$$p(y|x) = \prod_{i=1}^m p_i^{\delta_{yi}},$$

and its logarithm is

$$l(y|x) = \sum_{i=1}^m \delta_{yi} \log p_i,$$

where $p_m = 1 - \sum_{i=1}^{m-1} p_i$ is seen as a function of $(p_1, \dots, p_{m-1})$.

The FIM can be obtained with

$$g_{ij} = -\mathbb{E}_y[\partial_i \partial_j l(y|x)].$$

We have

$$\partial_j l(y|x) = \frac{\delta_{yj}}{p_j} - \frac{\delta_{ym}}{p_m},$$

then

$$\partial_i \partial_j l(y|x) = -\delta_{ij} \frac{\delta_{yj}}{p_j^2} - \frac{\delta_{ym}}{p_m^2}.$$

Since $\mathbb{E}_y[\delta_{yi}] = \mathbb{P}[y = i] = p_i$, we have

$$g_{ij} = \frac{\delta_{ij}}{p_i} + \frac{1}{p_m}.$$

Using the coordinates $\xi_i = 2\sqrt{p_i}$, it can be shown that the probability simplex equipped with the FIM can be *isometrically* embedded into the $m-1$ sphere of radius 2 equipped with the induced Euclidean metric from $\mathbb{R}^m$. The angle between two points (p_i) and (q_i) of the sphere with respective coordinates (ξ_α) and (ξ'_α) is

$$\cos\beta = \frac{1}{4} \sum_{\alpha=1}^m \xi_\alpha \xi'_\alpha = \sum_{i=1}^m \sqrt{p_i q_i}.$$

The Riemannian distance s between these two points is the arc length on the sphere:

$$s = 2 \arccos \sum_{i=1}^{m} \sqrt{p_i q_i}.$$

The geodesic connecting them is

$$\gamma_i(t) = \frac{((1-t)\sqrt{p_i} + t\sqrt{q_i})^2}{\sum_{j=1}^{m}((1-t)\sqrt{p_j} + t\sqrt{q_j})^2}.$$

4.5 Data foliation during training [Week 34]

I want to prove that the training set lies on the same leaf as claimed in [35]. More precisely, I want to prove that, when training a model with SGD, the training points converge to the same leaf when the number of iterations goes to infinity. This formulation is still vague.

Then, I want to understand what happens exactly around a training point in order to explain adversarial vulnerability.

My intuition is that we should use the "full information matrix", that is the information matrix with respect to the inputs x *and* the weights w. Indeed, given a training point x, the transverse distribution $(\ker G_x)^\perp$ is spanned by $\nabla_x \ln p^i(x, w)$, while SGD consists in moving in the w space in the direction of $-\nabla_w \ln p^{i(x)}(x, w)$ where $i(x)$ is the true label of x. So, what we want to understand is how $\nabla_x \ln p^i(x, w)$ changes when w becomes $w - \lambda \nabla_w \ln p^{i(x)}(x, w)$. I conjecture that the cross block of the full information matrix will give us the answer.

Moreover, I want to point out that a neural network can be seen as a composition of bilinear forms (followed by an activation function) where the bilinear forms' matrices are *fixed by the architecture of the network*. I don't know if this idea generalizes to any kind of neural networks, but it works at least for the most common one including MLP, CNN, ResNet, and probably RNN, Transformers. Maybe, the properties of these bilinear forms (that are entirely fixed by the architecture) will help us answer questions such as : under what conditions is the rank constant (and the distribution integrable), what happens to the geometry when we move through the network etc. In particular, with this formulation, x and w play symmetrical roles, so I don't see any reason for a foliation to never exist on the w space.

The kernel of the full information matrix defines a distribution. If this distribution is integrable, we obtain a foliation on the product manifold $\mathbb{R}^n \times \mathbb{R}^q$. This foliation induces a foliation on the submanifold $w = w_0$ which is the kernel foliation on the input space.

4.5.1 Notations

The notations are summarized in this diagram:

$$\begin{array}{lll} \mathbb{R}^n \times \mathbb{R}^q & \xrightarrow{N} \Delta^{m-1} \subset \mathbb{R}^m & \xrightarrow{l} \mathbb{R} \\ (x, \quad w) & \mapsto p & \mapsto l(y, p) = \sum_{i=1}^{m} \delta_i(y) \ln p^i, \end{array}$$

where $y \in \mathbb{R}$ can also be seen as a random variable whose distribution is the categorical distribution with parameter p.

4.5.2 Full information matrix

4.5.3 Neural network layer as bilinear form with activation

4.6 Isometric regularization [Week 36]

4.6.1 First idea

I conjectured that adversarial attacks are mainly due to a "discrepancy" between the metric $\overline{g}$ induced by the Euclidean metric in the data leaves (that we will also call the Euclidean metric for short) and the pullback

metric g of the FIM (that we will also call the FIM for short). Here, I want to give a precise meaning of this "discrepancy". What we want is that, given an Euclidean budget $\epsilon > 0$, the predicted probability around a training point does not change too much. In other words, we want the model N to be almost isometric around every training point, where the input space is endowed with the metric $\epsilon\overline{g}$.

More generally, we want the model to be locally isometric in an entire "input domain" (e.g., all the valid images). If we have enough training points well spread in the input domain that verify the local isometry property, then a Lipschitz assumption may be enough to guaranty the adversarial robustness in the entire input domain.

We know that the distribution $x \mapsto (\ker g_x)^\perp$ is spanned by $\{\nabla_x \ln p^i(x)\}$. Let x be a point of the input space $\mathbb{R}^n$. Let L_x be the leaf of x. Assume that L_x has dimension $m-1$ where m is the number of classes. We consider the restriction of the model N to L_x that we also call $N : L_x \to \Delta^{m-1}$. The model N is a local isometry if and only if $g_x(\nabla_x \ln p^i(x), \nabla_x \ln p^j(x)) = \epsilon\overline{g}_x(\nabla_x \ln p^i(x), \nabla_x \ln p^j(x))$ for $1 \leq i \leq j \leq m-1$. In the standard coordinates of $\mathbb{R}^n$, this is equivalent to $J^T(x)G_{N(x)}J(x) = \epsilon I$.

Why is information geometry useful here? Why not using the condition $J^T(x)J(x) = \epsilon I$ instead? I try to answer this question in the documents "Notions", "Regularization", and in section 5.8.

Maybe, the right transverse metric for the foliation of $(\ker G_x)^\perp$ is the Euclidean metric.

4.6.2 Looking for weaker conditions

In the document "regularization", we show that the isometry condition is equivalent to $JJ^T = kI$ where k is a constant that depends on the Fisher metric. With this condition, F becomes a local **partial isometry**.

With this condition, the leaves of the data foliation are flat i.e., their extrinsic curvature in $(\mathcal{X}, \overline{g})$ is zero. This can be shown by saying that the leaves must be totally geodesic for the Euclidean metric, and thus are composed of straight lines. The kernel foliation is also flat.

However, I still doubt this is true. If the leaves were flat (i.e., the second fundamental form is zero), then their curvature with the induced Euclidean metric is the same as the Euclidean metric, namely zero. But we impose that this curvature is the same of Δ^m which is nonzero! Hence, the second fundamental form is entirely determined by the curvature on Δ^m and is nonzero. The question is: how much degrees of liberty is left for the data or kernel foliations to encode any information for the classification?

Assume that the leaves are flat, then the model F can *only classify linearly separable datasets*! If the dataset is not separable, there is a **trade-off between achieving perfect robustness (flat foliations) and achieving good accuracy**. However, I want a method that can be both robust and accurate (and certifiable). I must find another weaker criteria to enforce robustness without harming the accuracy.

Similarity An isometry may be too strong. Another idea is to transform the Fisher metric with an isometry and an homothety ("similarity") in order to "shrink" the metric without destroying its information content.

Conformal model If we want to keep the criteria between the Euclidean metric and the Fisher metric, we can enforce that the model F is conformal (preserves angles but not distances). This condition is achieved by removing the conditions on the diagonal elements of JJ^T i.e., we ask that the rows of J be orthogonal but there isn't any condition on the norm of the rows anymore. How does this approach impact the robustness? And the accuracy?

Minimizing the maximum distance In the same vein as the "suppression of the highest eigenvalue", it might be possible to find a criteria, maybe taking into account the curvature, to minimize the maximum Fisher-Rao distance between two points of the ϵ_2-Euclidean ball.

Let us formalize this idea. Let $x \in \mathcal{X}$ and let $U \subseteq T_x\mathcal{X}$ such that $\exp(U) = B(0, \epsilon_2)$ where the ball is constructed with respect to the Euclidean metric, and exp is the exponential map with respect to the pullback metric.

Jacobi fields tell us how much a geodesic changes when the initial vector changes. When the only change is the length of the geodesic (and not the "direction"), that means that this initial vector corresponds to a maximum. Jacobi fields depends on the sectional curvature. My current understanding is that, if you consider only the eigenvalues of the metric, you assume that the curvature is zero, i.e., the eigenvalues and

eigendirections are the same in neighboring points. The true maximum of U depends on the curvature, which is a differential relationship. In the case of constant curvature, we may obtain simple ODEs with constant coefficients.

Another interesting quantity is the volume of the geodesic ball. Remember that a geodesic ball is the set of points that are images by the exponential map of tangent vectors in a given ball (with respect to the metric) of the tangent space. The variation of the volume of a geodesic ball depends on the Ricci curvature.

Partial isometry as a constraint Instead of using a regularization, we can enforce the partial isometry condition as a constraint, i.e., we minimize the cross entropy subject to F is a partial isometry. There exists algorithms to enforce this constraint (see optimization on the Stiefel manifold). However, according to [14], these algorithms are not scalable.

Transformation of the input space This is just an intuition. The aim is to take into account the "spatial" attacks and to address the discrepancy between $l2$ norm and "human dissimilarity measure". We may find a transformation of the input space to obtain new coordinates (maybe in smaller dimension) that better reflect the information content of each point. A kind of information-preserving compression that is optimal (i.e., as small as possible without destroying information). Then, we need to find a metric on this transformed space that reflects the "dissimilarity" between data points. When we think about it, there is no reason that the "pixel" space with the $l2$ metric is the right space/coordinates to use.

4.7 Existence of a local partial isometry [Week 40]

I realize that I didn't prove that a local partial isometry $F : \mathcal{X} \to \Delta^m$ even exists, where $F|_L \colon L \to \Delta^m$ is a local isometry for every leaf L of the data foliation.

A potentially devastating issue is that F may be a global isometry on each leaf. I need to know what are the conditions for a local isometry to be a global isometry. I read on a forum that if L is connected and complete, if Δ^m is connected, and if any two points of Δ^m admit a *unique* geodesic, then, if $F|_L$ is a local isometry, it is bijective (hence it is a global isometry).

If $F|_L$ is indeed a global isometry, I need to know if it is possible to fit together all these "bounded" leaves (which are all identical and look like pieces of sphere) into the data foliation. It may be impossible, thus F cannot exist!

If $F|_L$ is not necessarily a global isometry, I need to understand how is it possible to "glue together" pieces of a sphere into one leaf. If $F|_L$ is injective but not surjective, we get the same problem as above. If $F|_L$ is not injective, we have another behavior: since the leaves are connected (if I remember well from Molino), we must "reuse" a piece of the sphere several times to construct one leaf. It may be also impossible, thus F cannot exist!

Proposition 4.1 (From local to global isometry)**.** *Assume that:*

- $F : M \to N$ *is a local isometry.*
- M *is connected and complete.*
- N *is connected.*
- *Any two points of* N *can be joined by a* unique *geodesic parameterized by arc length.*

Then F *is bijective, and therefore* F *is a global isometry.*

Proof. **Injectivity**. Let $p, q \in M$ such that $p \neq q$. Let $I = [0, 1]$ and let $\gamma : I \to M$ be a geodesic such that $\gamma(0) = p$ and $\gamma(1) = q$. Cover $\gamma(I)$ by open sets (U_α) such that $F|_{U_\alpha} \colon U_\alpha \to F(U_\alpha)$ is an isometry. Since I is compact and γ is continuous, $\gamma(I)$ is compact. Thus, there is a finite covering (U_i). By taking smaller U_i if necessary, let $[t_i, t_{i+1}] = \gamma^{-1}(U_i \cap \gamma(I))$. □

Let us see if the assumptions of the proposition are verified for a typical classification task. The input space $\mathcal{X}$ is generally connected. It is not necessarily complete, for example if $\mathcal{X} = (0, 1)^n$. But, if I am not mistaken, the completeness assumption is only used to say that there always exists a geodesic between any

two points of $\mathcal{X}$, which is true for $\mathcal{X} = (0,1)^n$ (let us call this property (*)). If fact, it seems to me that completeness is not at all equivalent with (*). Strangely enough, this equivalence is claimed on some forums I read. As far as I understand, a manifold can be complete without verifying (*), e.g., take a compact subset of $\mathbb{R}^n$ that is not convex (in fact, I am not sure that a compact subset of $\mathbb{R}^n$ is complete!!!). A manifold can verify (*) without being complete, e.g., $(0,1)^n$. Anyway, the property (*) is generally verified for $\mathcal{X}$. Δ^m is connected and any two points of Δ^m can be joined by a unique geodesic (parameterized by arc length). Hence, for every leaf L, $F|_L$ is a global isometry.

So, can such a F exists? It may be possible that F exists but is not smooth on the whole of $\mathcal{X}$. There may be discontinuous points or non-smooth points at the boundaries of each leaf. This should be proven at some point.

Another question: is F entirely determined by a given foliation? Intuitively, the answer is yes. Since each leaf is in bijection with Δ^m, and each point of $\mathcal{X}$ belongs to some leaf, then the image of every point is fixed. Hence, this settles the question of the "degree of freedom" left in the kernel foliation: *there isn't any degree of freedom left!* However, there is still a degree of freedom in the *choice of the partial isometry* F. Some partial isometries may be more robust than other, or more accurate, or have a better balance between robustness and accuracy.

4.8 Distance between foliations [Week 40]

A criteria for the robustness of a model could be the distance between the current foliation of the model and a **Riemannian submersion** (i.e., each leaf is locally isometric to a given manifold). To achieve this, we must endow the space of foliations with a differentiable structure and a Riemannian metric. This criteria may be correlated with the robust accuracy and be used to evaluate the robustness of a model. The criteria can also be used to train a robust model (using a regularization term). We may derive this criteria for several output spaces:

- A space with constant positive curvature. This corresponds to classification when using the probability simplex, which is "compact".
- A space with zero curvature, i.e., $\mathbb{R}^m$, or a compact subset of $\mathbb{R}^m$.
- A space with constant negative curvature. For regression, this corresponds to the Poincaré half-space of normal distributions.

For similar ideas in complex analysis, see **moduli spaces**. For a given genus, a moduli space is a space of Riemann surfaces with a metric to compute distances between Riemann surfaces. In the complex setting the map are holomorphic, hence conformal, so it is easier. See the notion of **classifying space** for vector bundles. The space $\mathbb{R}^\infty$ of sequences with a finite number of nonzero terms is a classifying space of vector bundles, because the fiber of any vector bundles can locally be seen as a subspace of $\mathbb{R}^\infty$. See also **harmonic functions**. Eliot may also be looking at topological invariants: given a foliation, is there a continuous path to a Riemannian submersion?

4.9 Notes on integrability conditions of constant-rank distributions [Week 43]

I write this after listening to Emmanuel Gnandi. For a **constant-rank** distribution D, a necessary and sufficient condition for the integrability is given by Frobenius theorem: $[D, D] \subset D$.

Now, we are interested in the special case where $D = \ker g$ with g a pseudo-Riemannian metric[3]. It seems that for such metric, there is some sort of Levi-Civita connection ∇ (maybe not unique) with $\nabla g = 0$ (metric-compatibility) but not necessarily $T\nabla = 0$ (not necessarily torsion-less). The fact that $\nabla g = 0$ means that there exists a parallel transport that "preserves the metric" and thus, the distribution D has constant rank (I am not sure about that).

[3]In fact, a pseudo-Riemannian metric is a nondegenerate tensor field (but not necessarily positive). Here, we are interested in metrics that are positive but not necessarily nondegenerate. I don't know what if the name of these objects, so I will call them pseudo-Riemannian metrics.

Remember that the interior product i is defined for any covariant tensor T and any vector field X by $i_X T = T(X, \cdots)$. Thus we can define:

$$\ker g = \{X : i_X g = 0\}.$$

Frobenius theorem can be rewritten $i_{[X,Y]}g = 0$ for all $X, Y \in \ker g$. Using properties of interior product we have:

$$i_{[X,Y]}g = [\mathcal{L}_X, i_Y]g = \mathcal{L}_X i_Y g - i_Y \mathcal{L}_X g = -i_Y \mathcal{L}_X g,$$

using the fact that $i_Y g = 0$.

5 Problems

Summary of current problems [Week 40]

1. Partial isometry in each leaf. Can we certify this defense on the entire input domain? What about attacks outside the leaves? What about unrestricted attacks?

2. Is there a weaker criteria that is necessary and sufficient for adversarial robustness (at least in each leaf)? How to take into account the trade-off between robustness and accuracy? Is there really such a trade-off? Is there a trade-off between computational complexity and robustness?

3. Partial isometry may ensure constant rank. Is the rank really a problem for the existence of a foliation? What about singular foliations? Is the involutive property sufficient for the existence?

4. What if the output dimension is equal to or greater than the input dimension?

5.1 Existence of foliation: constant rank issue

How can we ensure that the local data matrix has constant rank (this is necessary for the foliations to exist)? If the rank is not constant, is there a singular foliation? If yes, what are its properties? What about "canonical stratification" if the rank is not constant? What are the conditions for a network to admit a kernel foliation (see the proofs in Tron et al., and in Molino)? Are we talking about regular foliations or singular foliations? In Tron et al 2022 and Grementieri and Fioresi 2021, they use the involution property, so does it apply only to regular foliations? What about the parallel leaves criteria?

According to Grementieri and Fioresi [35], the rank is constant only in the middle of the training, but then drop at the end of the training. They link this "loss of information plasticity" to the analysis of Achille et al. [36] where the authors show a similar phenomenon *but using the Fisher information with respect to the weights instead of the inputs*. However, I don't really see why the decrease of the trace of the FIM should be linked with the rank of the FIM. And even if this is true, I don't see why a decrease of the rank implies that *the rank is no longer constant!*

Whatever, I must explain the "loss of information plasticity" that we observe at the end of training (both in the data space and the parameter space). According to [36], a similarly phenomenon is observed in *biological brain!*.

1. During a first phase, the trace of the FIM increases. This is interpreted as the network quickly making lots of connections. During this phase, the test accuracy rises very quickly.

2. During a second phase, the trace of the FIM decreases. This is interpreted as "forgetting" or "compression" where redundant connections are eliminated and non-relevant variability in the data discarded (**this sounds suspiciously similar to "non-robust features"**). During this phase, the test accuracy continues to grow (which is counter-intuitive) but far more slowly than in the first phase.

Works related to [36] and computing tricks.

Grementieri and Fioresi claim that the distribution of the FIM over the parameter space is not integrable. They claim that it is "easy" to see *empirically* that this distribution is not integrable. I find this very unconvincing.

5.1.1 Some linear algebra

Consider the matrix $G(x) = J(x)G_{out}(N(x))J(x)$ where $x \in \mathbb{R}^n$, $\theta = N(x) \in \mathbb{R}^m$ and $n > m$. The matrix G_{out} has full rank everywhere because it is positive definite everywhere (it is a Riemannian metric).

1. Having constant rank is "equivalent" to having full rank, because having constant rank means that we can shrink the output space $\mathbb{R}^m$, i.e., get rid of useless coordinates, to obtain a full rank.
2. If J has full rank everywhere, then it is obvious that G has full rank m everywhere.
3. If J has constant rank $r < m$ everywhere, then G has rank r iff $\text{Im}\,(G^{out}J) \cap \ker J^T = \{0\}$.

Interesting classical result: for any matrix J (not necessarily square), we have $\ker J^T \perp \text{Im}\, J$. Let's prove that. Assume that we have $J : \mathbb{R}^n \to \mathbb{R}^m$ and $J^T : \mathbb{R}^m \to \mathbb{R}^n$. Let $x \in \ker J^T$ and $z \in \text{Im}\, J$. Hence there is y such that $z = Jy$. Then $\langle x, z\rangle = \langle x, Jy\rangle = \langle J^T x, y\rangle = 0$. It is certainly possible to prove that with the algebraic definition of the transposition. A consequence of this fact is that J and $J^T J$ have the same rank.

5.1.2 The Jacobian matrix has full rank is a generic property

We know that $\text{GL}_n(\mathbb{R})$ is open and dense in $\mathcal{M}_n(\mathbb{R})$. It may be possible to extend this result to non-square matrix where $\text{GL}_n(\mathbb{R})$ is replaced by the set of full-rank matrices. Then, it may be possible to show that, for an open and dense set of smooth functions, the Jacobian matrix has full rank at every point (or maybe on a set of points that is open and dense).

5.2 Existence of a foliation: dimensionality issue

Image and kernel foliations arise naturally for classification tasks since the number of classes is far smaller than the dimension of the input. However, it may not be the case for regression tasks, in which I am more interested. How does my theory stand if there aren't any foliation for dimensionality reasons, or more precisely, when the entire input space *is* the data leaf. In this case, adversarial attacks must be explained by moving in the data leaf! How can we explain noise since there aren't any noise leaves?

What happens if the output space has a higher dimension than the input space? Maybe, we can endow the image of the input space with the induced metric of the output space and then pull it back to the input space. Hence, we are in the case of input and output spaces with the same dimension. But is it always possible to endow the image of a smooth map with the induced metric (the image might not even be a submanifold).

5.3 Existence of a foliation: with respect to the parameters of the model

It is claimed by [35] that the distribution with respect to the parameters $w \mapsto \ker\{J^T(w)G_{N(w)}J(w)\}^\perp$ is not integrable because it is not involutive. This is due to the fact that, for a classical neural network, the parameters of one layer are multiplied with the parameters of the other layers. Hence, the Hessian of the output with respect to the parameters is not zero (while it is zero with respect to the input) and this prevents the distribution from being involutive.

However, we can see that the parameters of the same layer are not multiplied with each other. Hence, the Hessian of the parameters has a diagonal of zero blocks (one block for each layer). What happens if we restrict our scope to a subspace of parameters from the same layer, the other parameters being fixed? Then, the distribution is certainly integrable. What prevent the full distribution to be integrable is that, at each point w, it does not lie in one subspace. The directions of the distribution that cross several subspaces prevent the distribution to be integrable.

5.4 Compact leaves

We sometimes obtain compact leaves. Is a foliation with compact leaves necessary singular (think of the circles in $\mathbb{R}^2$ with a singularity at the origin)? Answer: **No**. For example, consider the foliation of the torus defined by straight lines with a chosen rational slope. All the leaves are compact and 1-dimensional. So the foliation is regular.

5.5 Rationals for using KL divergence and/or the Fisher-Rao distance to study adversarial attacks

Let us consider the classification setting. The output space Z is a $C-1$ dimensional manifold, where C is the number of classes. Let $p = (p^1, \dots, p^{C-1}, 1 - \sum_{i=1}^{C-1} p^i) \in Z$. The predicted class is $\arg\max p$. Now consider $\tilde{p} = p + \eta$ with $\sum \eta^i = 0$. We have that $\tilde{p}$ is an adversarial example iff $\arg\max \tilde{p} \neq \arg\max p$. Why would the KL divergence between p and $\tilde{p}$ be a good proxy for constructing adversarial examples? Given two perturbations η_1 and η_2, note that we have:

$$\mathrm{KL}(p||p+\eta_1) - \mathrm{KL}(p||p+\eta_2) = \sum p^i \ln \frac{p^i + \eta_2^i}{p^i + \eta_1^i}.$$

Consider the case $C = 2$ and $p = (0.51, 0.49)$. We impose the constraint $\|\eta\|_2^2 < 2.10^{-2}$. Consider $\eta_1 = (-0.1, 0.1)$ and $\eta_2 = (0.1, -0.1)$. Note that $p + \eta_1$ is an adversarial attack while $p + \eta_2$ is not. So, if the KL divergence is a good proxy, we would expect that $\mathrm{KL}(p||p+\eta_1) - \mathrm{KL}(p||p+\eta_2) > 0$. However, we have:

$$\begin{aligned}
\mathrm{KL}(p||p+\eta_1) - \mathrm{KL}(p||p+\eta_2) &= 0.51 \ln \frac{0.51 + 0.1}{0.51 - 0.1} + 0.49 \ln \frac{0.49 - 0.1}{0.49 + 0.1} \\
&= 0.51 \ln \frac{0.61}{0.41} + 0.49 \ln \frac{0.39}{0.59} \\
&\approx -2.10^{-4} < 0.
\end{aligned}$$

Maybe my example is not representative of the "typical" behavior of the KL divergence with respect to adversarial attacks. But it is easy to see that if you consider $\tilde{p} = (1 - \epsilon, \epsilon)$, then $\mathrm{KL}(p||\tilde{p})$ goes to infinity as ϵ goes to zero, while the predicted classes are always the same. On the contrary $\mathrm{KL}((0.5 + \epsilon, 0.5 - \epsilon)||(0.5 - \epsilon, 0.5 + \epsilon))$ goes to zero as ϵ goes to zero, while the predicted classes are always different.

What we are looking for is some kind of metric, or distance, or divergence d, verifying the *Class Separation Property*:

$$\text{for all } x_0, x_1, x_2 \text{such that } N(x_0) = N(x_1) \text{and } N(x_0) \neq N(x_2), \text{we have } d(x_0, x_1) < d(x_0, x_2). \tag{4}$$

The function d doesn't need to verify the triangular inequality, nor being symmetric, nor even verify $d(x, x') = 0 \Leftrightarrow x = x'$. However, we would like d to have some differentiability regularities and to be "easy to compute" given x and x'. If it is too hard, or impossible, for any d to verify Equation 4, then we would like to find a d that verifies Equation 4 for "typical" x_0, x_1, x_2, either with an "almost sure"-like property or an "open and dense"-like property.

A first idea is to look at Bregman divergences. There are defined as follows. Let $F : X \to \mathbb{R}$ be a strictly convex C^1 function, where X is a convex set. Let $q \in X$. The first order Taylor expansion of F at q is, for all $p \in X$, $F(p) = F(q) + \langle \nabla F(q), p - q \rangle + o(\|p - q\|)$. The Bregman divergence associated with F is:

$$D_F(p, q) = F(p) - F(q) - \langle \nabla F(q), p - q \rangle,$$

so we have $D_F(p, q) = o(\|p - q\|)$. We have:

- $D_F(p, q) \geq 0$ because F is convex.
- $D_F(p, q) = 0 \Leftrightarrow p = q$ if F is strictly convex, and $D_F(p, p) = 0$ for all p if F is convex.
- $D_F = D_G$ iff $F - G$ is an affine function.
- $D_F(p, q)$ is (strictly) convex in its first argument because F is (strictly) convex.
- For $\lambda \geq 0$, we have $D_{F_1 + \lambda F_2}(p, q) = D_{F_1}(p, q) + \lambda D_{F_2}(p, q)$.
- Let x be a random vector defined on X. We have $\arg\min_p \mathbb{E}[D_F(x, p)] =$ mean vector of X. (I don't know the precise statement of this result).

If $F(p) = -H(p) = \sum p^i \ln p^i$ is the negative of the entropy then $D_F = \text{KL}$.

Another idea is to look at f-divergences, and more specifically at α-divergences. These divergences are defied only for probability distributions. The KL divergence corresponds to the case $\alpha = 1$.

A divergence is defined as a function $D : X \times X \to [0, \infty]$ such that $D(x, x) = 0$ for all $x \in X$. The KL divergence is the only divergence on the categorical distributions that is a Bregman divergence and an f-divergence.

After more thinking, it doesn't seem to be a problem. If $p = (0.51, 0.49)$ then that means that the model is very uncertain and should answer "I don't know". If the model was well trained, it is very likely that the input data was just pure noise. The disturbing fact about adversarial attack is that the starting point has high confidence, the final point also has high confidence but *on another class*, and both points are close in some l_p norm. In this case, which is really representative of adversarial examples, the KL divergence between the two points is high.

5.6 Distribution of the data

This is a more general question concerning the framework. We often assume that there exists a distribution $D = (X, Y)$ that generates the data (more in "notions" about this assumption). Here, X and Y are random variables, and D is also a random variable whose distribution is the joint distribution of X and Y.

Using the typical abuse of notations, we can write $D = \mathbb{P}(X, Y)$. When we train a model, the model is learning $F(x) = \mathbb{P}(Y|X = x)$ the conditional distribution of Y given $X = x$. Hence F and D are linked by the relation $\mathbb{P}(Y = y|X = x) = \mathbb{P}(X = x, Y = y)\mathbb{P}(X = x)$.

In a classification task, $\mathbb{P}(Y|X = x)$ belongs to the family of categorical distributions, which is a parameterized family, and thus is suitable for the information geometry framework. However, we haven't any information about the distribution $\mathbb{P}(X)$, which may not belong to a parameterized family. However, the support of $\mathbb{P}(X)$ may be known, and we define $\mathcal{X}$ as the support of $\mathbb{P}(X)$.

Can we draw a link between the information geometry framework (for adversarial robustness or in general) and the Bayesian framework (as in variational density propagation)?

5.7 Adversarial robustness and the brain

The human brain isn't vulnerable to adversarial attacks. But we have seen little data in our life (compared to the biggest model of today), and no adversarial examples. So, *how is it possible that we are so robust?* According to talks given by neuroscientists, there are multiple differences between the functioning of the brain and the current framework of machine learning. First, the brain doesn't use backpropagation to learn. As a side note, there exists artificial neural networks that don't use backprop: they are called Extreme Learning Machine (only one hidden layer) but their generalization power is limited. Another difference is that there are "time components" in the functioning of the brain. For example, some biological neuron might be deactivated at one moment, and at specific time, it will spontaneously fire. There are some structural similarities between CNN and the visual cortex, but this is maybe the only real similarity between the brain and current machine learning models.

A first conclusion is that we should not rely too much on "comparing our models with the brain". Our current paradigm (backprop) is different from the brain but still very powerful. We should try to understand robustness in this paradigm. The comparison with the brain is yet another vast topic and we don't want to go in that (for now). We are "forced" to rely on backprop and we should iterate from that.

Another conclusion is that **the backprop paradigm is fundamentally flawed**. I think lots of people agree with that, but there is no better alternative. These critics are linked with the claim that neural networks don't really have concepts/don't really "understand" what they are doing. Neural networks are just looking for statistical correlations and patterns in the data. At first, I thought that this claim was wrong and that enough computational power will eventually lead to truly intelligent models. I thought that this claim has its root in some instinct that the human "soul" is somehow "special" (or even supernatural) and that an algorithm as stupid as backprop couldn't emulate the "soul". This is obviously a very bad intuition that is based on dumb superstitions. But there are other reasons to think that backprop is flawed. For example, in [10], the authors shows that neural networks are unable to separate two hyperspheres, but still achieve extremely high accuracy by "adding wrong numbers to a right result". This shows that neural

networks don't understand what they are doing. The presence of adversarial examples is a manifestation of this phenomenon. I wonder if understanding why backprop models are sensitive to adversarial perturbations cannot lead us to another paradigm to train models.

5.8 Several problems [Week 41]

1. Are there metrics in $\mathbb{R}^n$ that are flat but not equal to the Euclidean metric?
2. Is the Jacobian matrix of a local isometry semi-orthogonal?
3. Lipschitz constants are never mentioned in Riemannian geometry. Why?
4. Why using the FIM instead of any other metric e.g., the Euclidean metric?
5. What happens if we compute the Taylor series at order 2 of the relative entropy between two far points?

5.8.1 Flat metrics in the Euclidean space that are not Euclidean

I don't have a rigorous proof but here is my intuition. The curvature is the derivative (or second derivative?) of the metric. Maybe the connection is the first derivative and the curvature is the second derivative?

Let Q be a positive definite matrix that is not the identity I. In the standard coordinates of $\mathbb{R}^n$, define a metric g such that its matrix is Q at every point. This metric is "constant", thus its curvature is zero, i.e., it is flat. However, g is different from the Euclidean metric. Since the curvature is an isometry invariant, there exists an isometry of $\mathbb{R}^n$ that transforms g into the Euclidean metric **in another coordinate system that is not the standard coordinates**. But g and the Euclidean metric **are different**.

In Riemannian geometry, people tends to consider that things that are "equal up to an isometry" are indeed "equal", or that the difference doesn't matter. But this difference matters a lot for us, since we are working in specific coordinates.

5.8.2 Why using the FIM instead of any other metric?

As presented in "geometric robustness", the FIM has several exceptional properties:

1. It is the "infinitesimal generator" of a large family of entropies, including the Rényi entropies and of course the relative entropy (KL divergence). In particular, the relative entropy is the loss function used in classification machine learning.
2. Chentsov's theorem. The FIM is the **unique** metric that is invariant under transformation of the sample space by a sufficient (in French "exhaustive") statistic. In particular, diffeomorphisms are sufficient statistics (it is is easy to see that using the Fisher-Neyman factorisation). Hence, the FIM is invariant under reparametrization (or variable change) of the sample space.

However, these properties seem useless for the robustness that we are studying. Our criteria is:

$$\overline{B}(0, \epsilon) \subseteq B_g(0, \eta_g(x)),$$

where $\overline{B}(0, \epsilon)$ is the Euclidean ball in the tangent space of x, $B_g(0, \eta_g(x))$ is the ball induced by the chosen metric g (i.e., the pullback of the FIM), and $\eta_g(x)$ is the distance from $F(x)$ and the decision boundary in the probability simplex. Note that $\eta_g(x)$ depends on the chosen metric g. If this condition is respected, whatever the chosen metric, then the robustness is ensured. Then, why using the Fisher metric (and not a simpler metric such as the Euclidean metric)?

Eliot and Nicolas suggested a potential answer. The Fisher metric may be the "best" metric in the sense that $B_g(0, \eta_g(x))$ contains the most points. It is true that any metric would ensure robustness, but the Fisher metric would ensure the **best trade-off between robustness and accuracy**. This is equivalent to:

$$B_g(0, \eta_g(x)) \subseteq B_{\tilde{g}}(0, \eta_{\tilde{g}}(x)),$$

for "every" metric g, where $\tilde{g}$ is the pullback of the FIM.

	Predicted class	Predicted probability (%)
Original example x_0	5	98.10
FGSM example $\tilde{x}$	3	74.81
Projection on the data leaf x_T	3	70.57

Table 1: Predicted class and probability for the original example (Figure 2a), FGSM example (Figure 2d), and projection of the FGSM example on the data leaf (Figure 2c).

6 Experiments

From Xu et al. [3] §4.3.4: "The authors claim that the properties which are intrinsic to adversarial examples are not very easy to find. They also gave several suggestions on future detection works:

1. Randomization can increase the required attacking distortion.
2. Defenses that directly manipulate on raw pixel values are ineffective.
3. Evaluation should be down on multiple datasets besides MNIST.
4. Report false positive and true positive rates for detection.
5. Evaluate using a strong attack. Simply focusing on white-box attacks is risky."

6.1 Behavior of the FGSM attack with respect to the data leaf

We want to investigate how some classical attacks behave with respect to the data leaf. Are they in the data leaf? Are they far from it? At what angle? We begin with the FGSM attack.

6.1.1 Experiment setup

We use the MNIST dataset with LeNet (32 and 64 channels respectively in the two CNN layers and 128 units in the FC layer). This is the CNN architecture of the official MNIST example of the PyTorch framework[4]. We train the model with SGD, batch size 64, learning rate 0.01, and we use seed=1 for replication purposes.

As in [35], we choose the model from the 10th epoch for the following experiments. The reason is that, according to [35], the rank of the local data matrix shrank by the end of the training *and* the local data matrix has not longer constant rank over the input space. So, we must choose a partially trained model in order to ensure that the rank is constant. We find this explanation very unconvincing, but for now we follow the advice by choosing a partially trained model.

The FGSM attack is implemented using this tutorial[5]. All the attacks will be performed on examples taken from the test set of MNIST. Figure 1a shows several FGSM examples for various attack budgets while Figure 1b of the model over the test set when the test examples are replaced by FGSM examples for increasing level of attack budget.

6.1.2 Projecting FGSM example on the data leaf

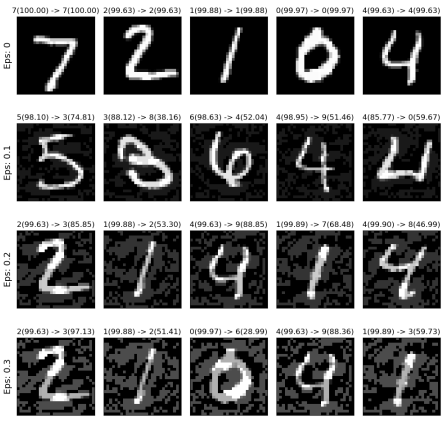
We use Algorithm 1 from [35] to find a horizontal path between each image x_0 and its adversarial example $\tilde{x}$. If the algorithm doesn't converge (i.e., the adversarial example is not on the data leaf), we use the last iterate x_T as an estimate of the Euclidean projection of the adversarial example $\tilde{x}$ on the data leaf.

We conduct this experiment on an example from the test set shown in Figure 2a.

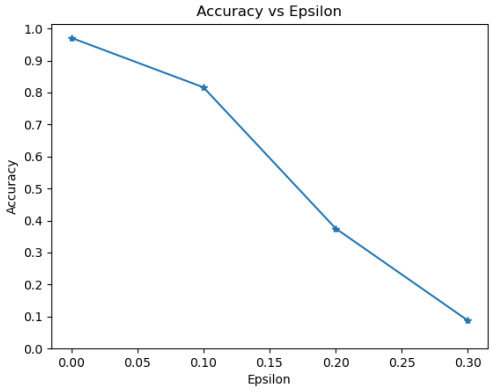
As shown in Table 1, the projection of the FGSM example on the data leaf is also an adversarial example. Now, we compute various distances and angles to see how the adversarial examples behave with respect to the data leaf. For example, the Euclidean norm $\|\tilde{x} - x_T\|_2$ gives us the l_2 distance between $\tilde{x}$ and the data leaf. The angle between $\tilde{x} - x_T$ and $(\ker G_{x_T})^T$ can be obtained as follows. Let $v = (\tilde{x} - x_T)/\|\tilde{x} - x_T\|_2$. We have $\cos\theta = \sqrt{\sum_{i=1}^{k}(v^T e^i)^2}$ where (e^i) is an orthonormal basis of $(\ker G_{x_T})^T$.

[4] https://github.com/pytorch/examples/tree/main/mnist

[5] https://pytorch.org/tutorials/beginner/fgsm_tutorial.html



(a) Each row corresponds to a certain attack budget in the l_∞ norm (denoted "Eps"). Five examples are given for each budget level. On top of each example, we provide the prediction of the model on the original test example and on the adversarial example along with the predicted probability. Note that the first row contains only original test examples since the budget is set to 0.



(b) Accuracy over the test set when the test examples are replaced by FGSM adversarial examples for increasing level of attack budget.

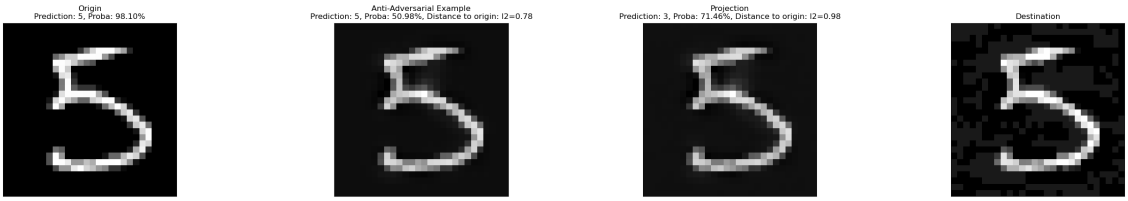
Figure 1: FGSM attack

	l_2 norm	l_∞ norm
$d(x_0, \tilde{x})$	2.04	0.100
$d(x_0, x_T)$	1.03	0.174
$d(x_T, \tilde{x})$	1.75	-

Table 2: Distances between various examples.

Angle between the data leaf and:	
$\tilde{x} - x_0$	62.44
$\tilde{x} - x_T$	88.41

Table 3: Angles in degrees between the data leaf and various vectors.



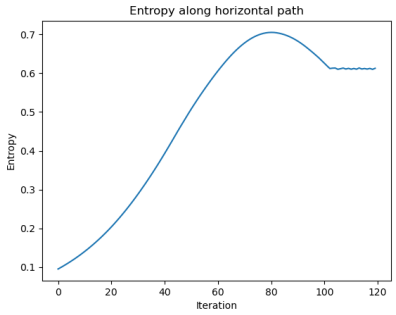
(a) Original image from the test set

(b) Anti-adversarial example

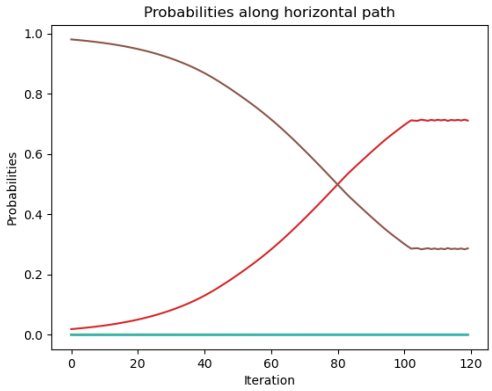
(c) Projection of the FGSM example on the data leaf

(d) FGSM example

Figure 2: Original example, anti-adversarial example, projection on the data leaf, and FGSM example. An anti-adversarial example is characterized as follows: the model is unsure about how to classify the example despite being clearly recognizable by humans.



(a) Entropy along the horizontal path



(b) Probabilities along the horizontal path

Figure 3: Entropy and predicted probabilities of each class with respect to the iterations of Algorithm 1 [35]. Note that the x-axis is the iteration not the distance. After the algorithm converges at the $\sim$ 100th iteration, the entropy and probabilities are constant because the distance between x_0 and the current iterate does not increase anymore.

As we can see in Table 2, the projection x_T is closer to the original image x_0 in l_2 norm but farther in l_∞ norm. In Table 3, we can check that the angle between the data leaf and $\tilde{x} - x_T$ is (almost) 90° as expected.

6.1.3 Anti-adversarial example

As conjectured in the paragraph 4.3.1, we check experimentally that the entropy reaches a maximum along the horizontal path connecting x_0 to x_T in the data leaf. When the entropy is maximum, the model is uncertain about the right class despite the image being easily classified by humans. We call these images *anti-adversarial examples* (Figure 2b). The entropy and probabilities along the horizontal path are presented in Figure 3.

TBD

- Do the experiments on multiple examples.
- Test the following attacks: PGD, OSSA. Then implement these attacks: C&W, DeepFool and maybe OSTCM, BIM (these last two might be very similar to FGSM and PGD). We may try FoolBox.
- Complete Table 2 with the Riemannian distance $d(x_0, x_T)$. To do so, we need an algorithm for the Riemannian distance (geodesic) along submanifold.

- Entropy, prediction, probabilities: origin-projection along the Euclidean path, origin-destination along the Euclidean path.
- Use the mean trace of $G(x, w)$ as a proxy for the rank and choose an epoch where the trace is the same as at the beginning of the training (or prove that this is meaningless).
- Run attacks on the training set to see if there is any difference as compared to the test set (probably not).
- Experiment with other datasets: CIFAR, ImageNet, and more.
- Comment the code!
- Why does FGSM work using only one linearization?

6.2 Moving in the data leaf along curves whose velocities are eigenvectors

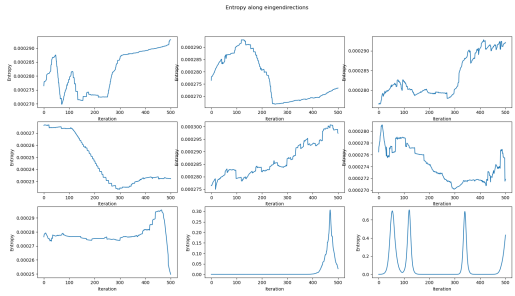
We are going to test this conjecture: *if we move along the data leaf by following eigenvectors of the local data matrix, then we will obtain an adversarial example of a certain class. Since there are (at most)* $m-1$ *independent eigendirections (where* m *is the number of class), we may obtain a targeted attack for each eigendirection.* We use the Algorithm 1 with $T = 500$ and $\alpha = 0.05$. With MNIST we have $m = 10$. The initial image x_0 is shown in the first row of Figure 5b.

Algorithm 1 Paths along eigendirections

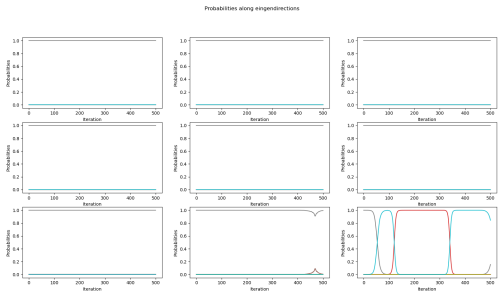
Require: initial point x_0, max iteration T, step size α, number of classes m
  $g \leftarrow localDataMatrix(x_0)$
  $v_1, \dots, v_{m-1} \leftarrow Eig(g)$ ▷ the $m-1$ unit eigenvectors associated to the $m-1$ highest eigenvalues of g in descending order
  for $i = 1; i \leq m-1; i++$ **do**
    $d_i \leftarrow v_i$
    $x_i \leftarrow x_0 - \alpha d_i / \|d_i\|_2$
  end for
  $k \leftarrow 1$
  while $k \leq T$ **do**
    for $i = 1; i \leq m-1; i++$ **do**
      $g \leftarrow localDataMatrix(x_i)$
      $v_1, \dots, v_{m-1} \leftarrow Eig(g)$
      if $v_i^T d_i < 0$ **then**
        $d_i \leftarrow -v_i$
      else
        $d_i \leftarrow v_i$
      end if
      $x_i \leftarrow x_i - \alpha d_i / \|d_i\|_2$
    end for
    $k \leftarrow k + 1$
  end while

We tested several variations for Algorithm 1:

- Instead of the i-th curve being associated to the i-th largest eigenvalue, we select the eigendirection that is most closely aligned with the previous direction. This is done with $j \leftarrow argmax(|v_1^T d_i|, \dots, |v_{m-1}^T d_i|)$ then $\tilde{d} \leftarrow v_j$ and $d_i \leftarrow \pm\tilde{d}$.
- The condition for choosing $d_i \leftarrow \pm\tilde{d}$ can be a global condition instead of a local one: compute $\tilde{x} \leftarrow x_i - \alpha\tilde{d}/\|\tilde{d}\|_2$ and if $\|\tilde{x} - x_0\|_2 < \|x_i - x_0\|_2$, then $d_i \leftarrow -\tilde{d}$ else $d_i \leftarrow \tilde{d}$, i.e., we want the direction that increases the Euclidean distance to the original point.



(a) Except for the two highest eigenvalues (bottom middle and right), the entropy stays almost constant and close to zero. For the highest eigenvalue (bottom right), we see three spikes corresponding to class changes (and it seems to be reaching a fourth spike). For the second highest eigenvalue (bottom middle), we have a smaller spike that does not lead to a class change.



(b) Except for the two highest eigenvalues (bottom middle and right), the probabilities stays constant with the probability of class "7" almost equal to one and the others almost zero. For the highest eigenvalue, we see three class changes (from "7" to "9", then from "9" to "3", then from "3" to "9").

Figure 4: Entropy and probabilities along paths whose velocities are eigenvectors of the local data matrix. The associated eigenvalues are sorted with the smallest one in the top left corner and the highest one in the bottom right corner (the increase is along row first).

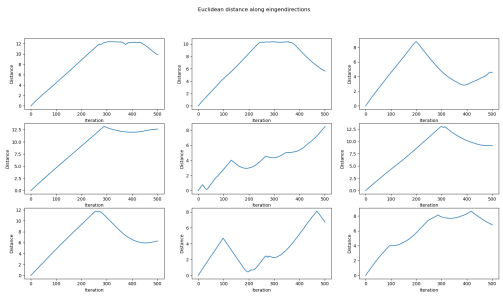
- Another update for x_i with a plus sign instead of a minus sign: $x_i \leftarrow x_0 + \alpha d_i / \|d_i\|_2$. It seems that sometimes, going in one direction doesn't lead to any class change, while going in the opposite direction does.

Now, looking at the results, we can conclude that the conjecture is false. We only obtain class changes in the eigendirection associated to the highest eigenvalue. We note the following observations:

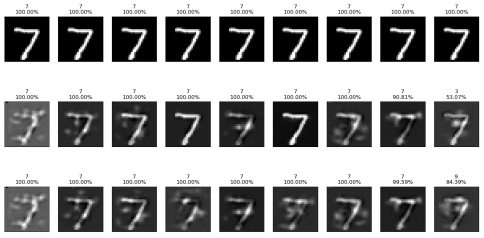
1. This is not an adversarial attack because the perturbation is clearly visible.
2. We observe three class changes that quickly follows each other, and a fourth was going to happen when we reached the last iteration.
3. The area of high extrinsic curvatures are not aligned with the area of high uncertainty (and class changes).
4. The second highest eigenvalue seemed to start a class change but finally went back to the original class.
5. All other eigendirections keep the same class with high confidence and low entropy.
6. The extrinsic curvature seems low around the original point but becomes higher farther from it.
7. The distance curves seems almost piecewise linear, as if the extrinsic curvature was generally low with small areas of high extrinsic curvature.
8. The model has a high confidence despite the images being very noisy.
9. Some images seems to contain information from other digits. In the last row of Figure 5b, the third, fourth and sixth images seem to contain the shape of a "5". The fifth image contains an horizontal bar in the "7" that wasn't present in the original image. And the last image contains the shape of a "9" which is coherent with it being classified as a "9".

TBD

- Compute the extrinsic curvature of the data leaf.



(a) For the first iterations, the distance increases almost linearly, indicating a low extrinsic curvature. Then, we observe several sharp slope changes, indicating areas of high extrinsic curvature.



(b) Each column corresponds to a different eigendirection. The first row is the original image. The second row is the image obtained at the point with maximum entropy. The third row is the final image obtained after N iterations. Only the highest eigenvalue (right most column) leads to class changes, but these are not adversarial examples because the perturbation is visible. For the other eigendirections, the model predicts the original class "7" with very high confidence despite the perturbations being clearly visible.

Figure 5: Distance along paths whose velocities are eigenvectors and examples of images along these paths.

6.3 Comparing models trained on original MNIST vs robustified MNIST

How do the FIM / the leaves change when using different datasets vs. different models? We can guess that different datasets (of the same underlying distribution) will lead to different FIM, while different models on the same dataset will lead to similar FIM. In particular, how does the FIM change when using a classical dataset vs. a robustified dataset?

6.3.1 Training a robust model with adversarial training

Using the same experiment setup as in subsection 6.1, we train a robust model using this code[6]. The robust optimization criteria is approximated by choosing a random starting point in a l_∞ ball of radius $\epsilon = 0.3$, then performing 40 iterations of PGD with step size 0.01. PGD is an iterative attack where, at each iteration, we perform a small gradient attack (FGSM attack in our case) then we project the adversarial example in the l_∞ ball of radius $\epsilon = 0.3$ and in the acceptable range of inputs (between 0 and 1 in our case).

In the following, we denote by N_{std} the model trained on the standard MNIST dataset $\mathcal{D}$ and N_{adv} the model trained by adversarial training.

6.3.2 Constructing a robustified MNIST dataset

We use the method described in [13]. We want to construct a robustified dataset $\mathcal{D}_r$ with the same size as $\mathcal{D}$ using a one-to-one function $\mathcal{D} \to \mathcal{D}_r, x \mapsto x_r$. The image x_r is built from x using an iterative process. Let g_{adv} be the composition of all layers until the penultimate layer of $N_{adv} = f_{adv} \circ g_{adv}$ (the last layer f_{adv} being a softmax regression layer i.e., $f_{adv}(z) = \text{softmax}(Wz + b)$). We minimize the loss

$$l(z) = \|g_{adv}(z) - g_{adv}(x)\|_2,$$

using normalized gradient descent i.e., the gradient is normalized and the iterates are clamped between 0 and 1 to be admissible images. The initial image is uniformly sampled from $\mathcal{D}$. For each image x, we perform 100 iterations with step size 0.2. We define x_r to be

$$x_r = \arg\min_z l(z).$$

[6] `https://github.com/ylsung/pytorch-adversarial-training`

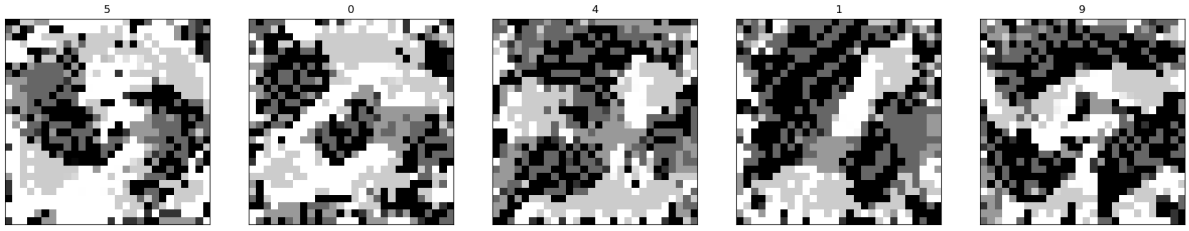


Figure 6: Examples of images from the robust MNIST training set. The true label is indicated above each image.

Model	Standard accuracy	Adversarial accuracy
N_{std}	97.11 %	16.13 %
N_{adv}	97.72 %	85.61 %
N_r	73.96 %	38.30 %

Table 4: Standard and adversarial accuracy for the three models

Figure 6 shows several examples from the robustified training set. It is difficult (while not impossible) for a human to recognize the correct label of these images. Nonetheless, they contain enough information to reach a decent accuracy while being more robust than the standard model N_{std}.

Now, we trained a third model N_r on $\mathcal{D}_r$ but without any adversarial training. The hyperparameters are the same[7] as described in subsection 6.1. The three models N_{std}, N_{adv} and N_r are tested on the standard MNIST test set composed of 10,000 images. To measure the robustness of each model, The three models are also tested on adversarial example of each image of the test set. The adversarial samples are obtained using PGD with $\epsilon = 0.3$, 20 iterations, step size 0.01 and l_∞ norm.

As shown in Table 4, N_r has a higher adversarial accuracy than N_{std} but a lower standard accuracy.

Remark. During my first attempt to create the robustified MNIST, I made a mistake and used a randomly initialized model instead of the adversarial trained model. The obtained images look like random noise (not at all like Figure 6). However, the model trained on this dataset reaches very similar performances as N_r, as well as in accuracy on the normal test set, *and accuracy on the adversarial test set.* This is very surprising and I cannot explain this phenomenon for now.

6.3.3 Exploring the data leaf of the three models

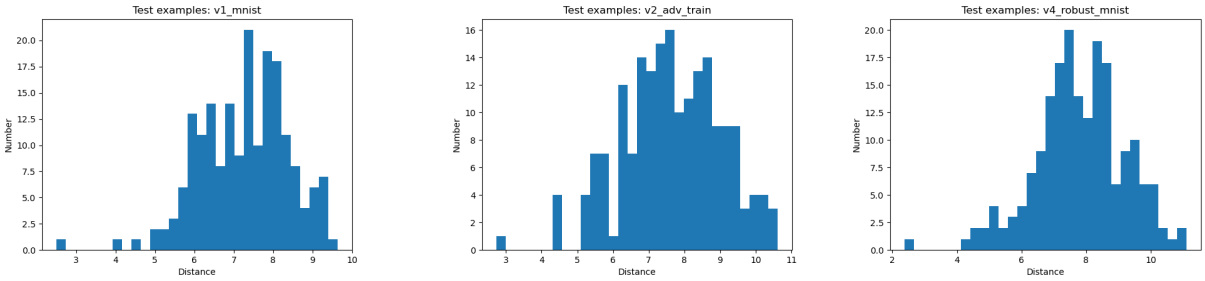
We want to test the following conjectures:

- The training *and* test data lie on the same leaf (the data leaf). This is less true for the test set than for the training set.
- The adversarial examples are not on the data leaf.
- We can distinguish between the three models N_{std}, N_{adv} and N_r using only the distributions of the training, test, and adversarial examples in the foliation.

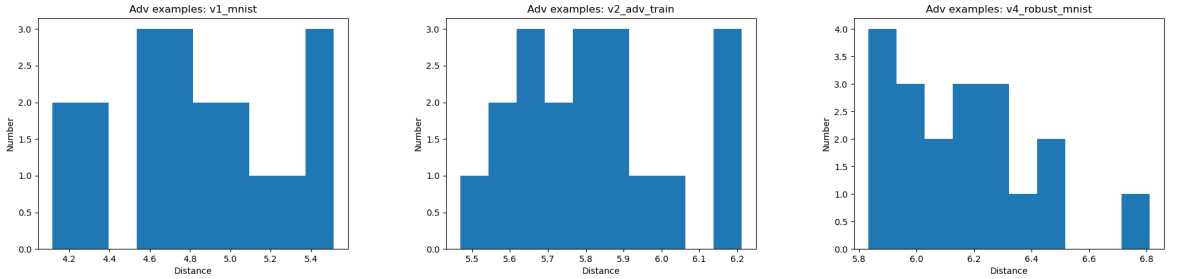
In order to test these conjectures, we randomly select 20 images from the training set and 20 images from the test set. For each of these images, we craft an adversarial attack using PGD with the l_∞ norm, 100 iterations and a budget $\epsilon = 0.3$. Then, for each model, we conduct two experiments:

1. For each pair of original (i.e., not adversarial) images, we compute the horizontal path using Algorithm 1 from [35] from the first image to the second image. The Euclidean distance between the last iterate

[7] with the exception that we choose the model after the second epoch instead of the tenth because the model seems to overfit the training set after the second epoch. This phenomenon was not observed for the other models.



(a) Original images using N_{std}. (b) Original images using N_{adv}. (c) Original images using N_r.



(d) Adversarial images using N_{std}. (e) Adversarial images using N_{adv}. (f) Adversarial images using N_r.

Figure 7: Histograms of distances using 20 images from the **test** set.

and the second image is seen as the distance between the second image and the leaf of the first image[8]. If all images are in the same leaf, we expect this distance to be small.

2. For each original image, we compute the horizontal path between this image and its associated adversarial image. Once again, the Euclidean distance between the last iterate and the adversarial image gives us the distance between the adversarial image and the leaf of the original image. Here, if the adversarial images are not in the data leaf, we expect the distance to be larger than in the former experiment.

The horizontal path are computed with a step of 0.1, and a maximum number of iterations of 200. To avoid excessive computation time, we use a robust stopping criterion. If the loss function is f (i.e., the distance between the current iterate and the destination) then the criterion is:

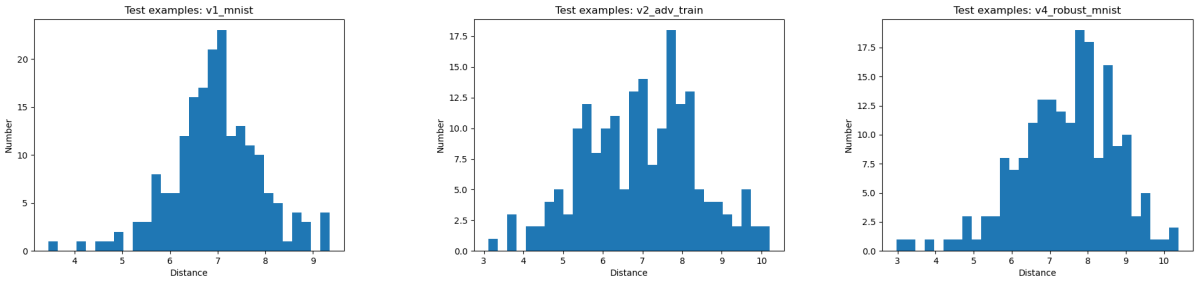
$$\frac{k|f(x_k) - f(x_{k-1})|}{|f(x_k)|} < \eta$$

where $\eta = 0.001$ is a tolerance.

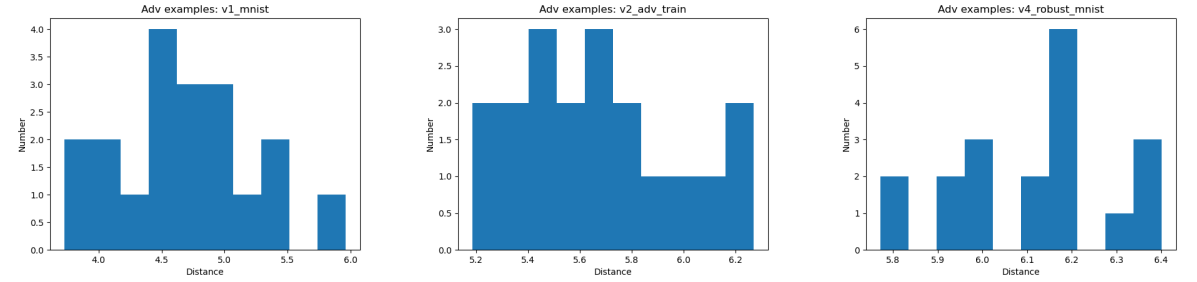
In Figures 7 and 8, we plot the histograms of the distances for each experiment. As we can see, *all the conjectures are false*. The distances for the original images are not close to zero and there is no difference between the training and test sets (first conjecture is false). The distance for the adversarial images are actually *smaller* than for the original images (second conjecture is false). The three models generate the same distributions for the original images as well as for the adversarial images (third conjecture is false). These results seem to show that the data leaf *does not even exist* which is in contradiction with [35] where the existence of the data leaf was the main result.

In [35], the authors use at least 5000 iterations, up to 10000 iterations, to construct the horizontal paths. Maybe, our horizontal paths were stuck in a local minimum. Because of the computation time, we cannot conduct these experiments with 5000 iterations. However, we can compute the horizontal path for a small number of examples. In Figure 9, we plot the evolution of the distance between the current iterate and the

[8]Note that this relation is not symmetric: this is not the same as the distance from the first image to the leaf of the second image. To avoid excessive computation time, we do not compute this other distance.

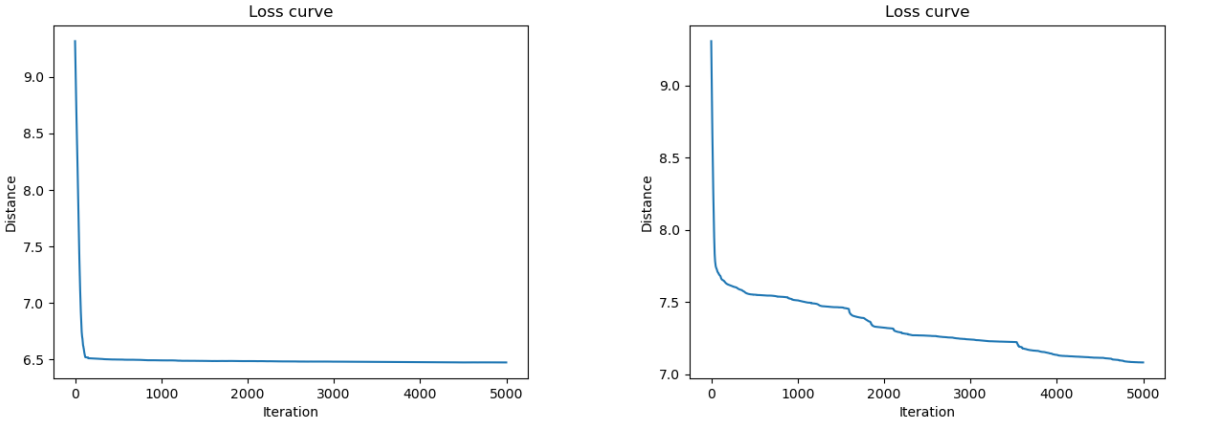


(a) Original images using N_{std}. (b) Original images using N_{adv}. (c) Original images using N_r.



(d) Adversarial images using N_{std}. (e) Adversarial images using N_{adv}. (f) Adversarial images using N_r.

Figure 8: Histograms of distances using 20 images from the **training** set.



(a) Loss function using N_{std}. (b) Loss function using N_{adv}.

(c) Horizontal path using N_{std}. (d) Horizontal path using N_{adv}.

Figure 9: Loss function and horizontal path between two original images of the training set using N_{std} and N_{adv} with 5000 iterations. The first image is the origin, the last image is the destination, the five central images are sampled regularly along the horizontal path. Above each image, we indicate the iteration and the predicted class.

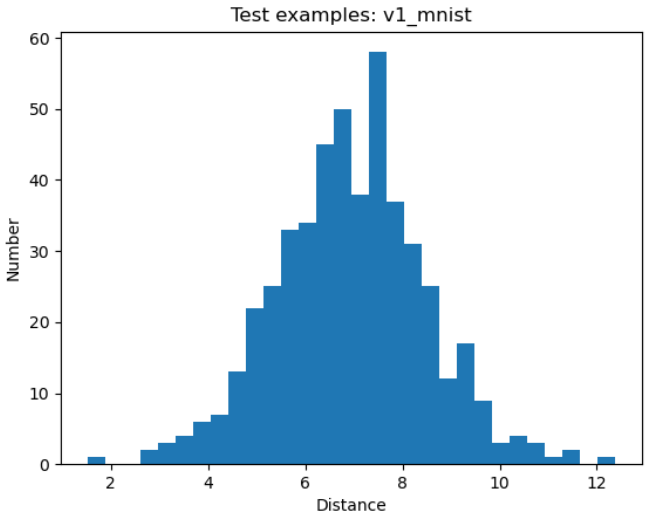


Figure 10: Histogram of distances between images of the **same class** using N_{std} and 100 images from the training set.

destination for 5000 iterations. If we were stuck in a local minima, the algorithm does not seem to escape from it with more iterations. Moreover, the actual paths (also shown in 9) indicate that it is impossible to reach the destination. This is especially true for N_{adv} where it seems that we cannot modify the image outside of the digit when moving along the leaf.

Now, we can conjecture that there isn't a single data leaf, but one data leaf for each class. To test this conjecture, we consider 100 images from the training set. As in Figure 8, we plot the histogram of distances using N_{std} **but considering only the horizontal paths between images classified in the same class**. If images of the same class are in the same leaf, we would expect the distances to be close to zero. As we can see in Figure 10, this is not the case.

TBD

- Compute some statistics.
- Horizontal paths for partially trained models
- Check that a dataset robustified on a random model is more robust than a standard dataset. What happens if we use the standard model N_{std} to robustify the dataset?
- Consider the adversarially trained model N_{adv}. Move an image into the leaf of another image *through the kernel of the FIM*. The predicted class should not change. However, it seems that two images of the same leaf can only differ in the "digit pixels" (i.e., the features) of the training image associated to the leaf. Since the Euclidean projection obtained with the horizontal path has a different class, both images are different. Does it mean that the extrinsic curvature is not negligible?

6.4 Geometry of the dataset

Mean Euclidean/Riemannian/l_p distance in the training/test sets. In each class

TBD

- Closest (Euclidean/Riemannian/l_p) data point of an adversarial example: of the same prediction, of the same "true" class.
- With other l_p distances. With the test set.

6.5 Out-of-distribution data with respect to the data leaf

TBD

- Obtain out-of-distribution data, from other datasets, or by brightening, rotating MNIST images.
- See where out-of-distribution data lie with respect to the data leaf.

6.6 Isometric regularization

Histograms of regularization values for successful vs unsuccessful attacks. I hoped that successful attacks will correspond to higher regularization value. Unfortunately, the distributions of regularization values for successful and unsuccessful attacks look the same.

Check that the Jacobian matrix is "orthogonal". This is done by checking that the singular values of J are close to the same value.

TBD

- Check the eigenvalues of G_x.
- Show that the method works on a toy problem (e.g., OR, XOR). I may find the weight of a small network myself. The network should exist, be robust, and be accurate.
- Evaluate the method: other datasets, other defenses, other attacks (C&W). Implement the fooling ratio.
- Use cross validation to choose the regularization parameter.
- Does the computational graph of Pytorch contain the gradient with respect to the input?
- Test a weaker condition: F is conformal (it preserves angles but not distances), i.e., the rows of J are orthogonal and they all have the same norm (however, contrary to an isometry, the norm is free): $JJ^T \propto I$.
- Compute the extrinsic curvature and compare a vanilla model with a regularized model.

6.7 Jacobian regularization

To be published in IEEE Transactions on Neural Networks and Learning Systems.
In green → what has been done

6.7.1 Implementation

Toy model → synthetic dataset (something like OR/XOR) with a small neural net that:

- exists,
- is perfectly robust,
- is accurate.

This will motivate the approach.
Datasets

- MNIST, CIFAR-10.
- Maybe: CIFAR-100, STL-10, SVHN, Fashion-MNIST, GTSRB, Tiny ImageNet.

Attacks (using the torchattacks package).

- PGD with l_2 norm and with l_∞ norm (iterations of FGSM)
- DeepFool
- C&W. Hyperparameters need to be tuned.

Defenses

- Baseline (no defense)
- Adversarial training (with FGSM)
- Defense distillation
- Suppressing largest eigenvalue (regularization method with similar justifications)
- Parseval networks (similar regularization method but different justifications)

Evaluation metrics

- Proportion of perturbed images that are correctly classified among the images that were already correctly classified (when not perturbed) $\rightarrow$ this is equivalent to the fooling ratio
- It is also possible to use the signal-to-noise ratio C/N0 instead of the budget.
- Report also the accuracy on the test set, since there is (seemingly) a trade-off between accuracy and robustness.
- Report also the training time, since adversarial training will very likely be more robust, but slower to train.
- Use Wandb.

6.7.2 Experiments

Plots

- Plot the robust accuracy with respect to the attack budget using test data. For MNIST, budget between 0 and 0.2.
- Robustness to Gaussian noise. Plot the accuracy with respect to the variance of the noise.

Training

- Do not use the "increasing η method". Test several fixed eta (like 10^{-n}). We may have a specific η for each dataset.
- We may see η as a learnable parameter.

Problems

- "Cross-validation" to choose η. Cross˙validation consists in dividing the training set in n chunks, then train on $n-1$ chunks and measure whatever you want to measure on the nth chunk (typically the accuracy). Then redo that for all possible combinations. By doing this, we obtain a measure of the accuracy that has nice statistical property. In order to optimize for a hyperparameter, you have to define a grid of values and perform cross-validation on each value (which is very long). The statistical property of cross-validation ensures that the value with highest accuracy will certainly be the best one. Once, the hyperparameters are chosen, retrain everything on the entire training set.
- "Expressivity" of the model. Seems to be linked with the accuracy/robustness trade-off? In the Parseval network paper, the authors want to measure the "use of capacity" of the network. They use "local covariance matrix". Other measure of expressivity?

6.7.3 Improvements

Done

- Check the largest singular value of the Jacobian matrix of some training points with respect to the bound. Results: the bound is respected for most points.
- Merge add_weights_and_biases with main. Create 2 files: one for training and one for testing (avoid too long files).
- $\rho(x) = 1 - \sqrt{F^{m+1}(x)}$. Use smaller ϵ. Use the true $\lambda_{max}(\tilde{J}_x^2 \tilde{J}_x)$.
- Remove $1/n$.
- New plots, per batch: $\|J\|$-bound, α (without η), H.
- Train, only 10 epochs : 2 baselines (with different seed), 3 jacobian (with different ϵ), 1 distilled, 1 parseval, 1 suppress max eigenvalue, 1 only jacobian regularization, 3 isometry (different ϵ), 1 adv train.
- Plot the robust accuracy of all 13 models: PGD l_∞ from 0 to 0.25, step of 0.025 ; GN from 0 to 0.15, step of 0.015.
- How to use pip (to install torchattacks)[9], how to git clone[10].
- Test the 13 models with AutoAttack (see Croce and Hein, "Reliable evaluation ...", 2020).
- Multiple runs, and report runtime.
- CIFAR-10.

Next

- Polytope closest point for δ (in Euclidean distance, and maybe Riemannian distance).
- See [37] for an efficient approximation of the Jacobian.

Optional

- It seems that composing two softmax somehow obfuscate the gradients of the attack. To test this hypothesis, craft attacks on a normal model and transfer them on a double softmax model. We expect the attacks to be successful. We may also compare the weights of two models with the same architecture to see if they converged to the same critical point or not.
- Jacobian norm visualization: use the l_∞ norm (highest element of the matrix) instead of l_2 norm (spectral norm). If the Jacobian was so uninformative, how is PGD working?
- Use residual connections (ResNet). A paper claims that ResNets have a smoothing effect on the loss landscape ...
- In order to detect potential errors in the code, check that $\tilde{g}_x(X, X) \leq \frac{\delta(x)^2}{\epsilon^2} \overline{g}_x(X, X)$ whenever $\|\tilde{J}_x\|_2 \leq \frac{\delta(x)}{\epsilon \rho(x)}$. It can be done with Monte Carlo sampling.
- Similarly, test the inclusion of geodesic balls. It can be approximated by testing the robustness to noise. Maybe, the method is robust to Gaussian noise, but not to PGD attack.
- Maybe MNIST is so simple that the baseline model is already robust and the regularized model harms accuracy without improving robustness. I can test another more complicated dataset.

[9] https://urllib3.readthedocs.io/en/1.26.x/advanced-usage.html#https-proxy-error-http-proxy
[10] https://github.com/lshigarrier/geometric_robustness.git

- To detect errors in the code, try a 2-dimensional linearly separable dataset to see if the regularized model works (it should be identical to the baseline model here).
- Implement TRADES and FIRE (variants of adversarial training).
- Test the 13 models with C&W.

Improve the computation of the Jacobian matrix

- Find other barrier functions.
- Replace the Frobenius norm by Hölder's inequality:

$$\|A\|_2^2 \leq \|A\|_1 \|A\|_\infty = \left(\max_j \sum_i |a_{ij}|\right)\left(\max_i \sum_j |a_{ij}|\right).$$

- See [37] for an efficient approximation of the Jacobian.
- We may use the trace to upper-bound the largest eigenvalue.
- Approximation of the largest eigenvalue / singular value: see the PyHessian paper for an alternative to Lanczos algorithm, or power iteration.
- See [38]. Is there an efficient way to access the Jacobian matrix (gradients with respect to the inputs) in PyTorch? It seems possible to exploit the fact that PyTorch compute the gradients with respect to the weights to speed up the computation of the Jacobian, but how? Is there something to do with the "computational graph"?

Theoretical improvements

- Fix a robust accuracy value. Find the highest accuracy for this fixed robustness. We may find a theoretical result in the vein of Neyman-Pearson test. At least, we may certify the (relative) robustness on the training set by allowing only a fixed proportion of training point to violate the robust bound. This sounds like integer programming with $0, 1$ values
- Replace the FIM with another metric (whose eigenvalues are the distances to the decision boundary?) or replace the categorical distributions with another family. What about softmax temperature? When temperature goes to infinity, softmax goes to max.
- Find an exact formulation of the robustness condition (taking into account the curvature), then find a computationally tractable method. What about Hessian matrix in addition to Jacobian matrix ? ...

7 Actions

7.1 Questions

- Does a transverse foliation always exists? What are the properties of a transverse foliation with respect to the properties of the original foliation (e.g., if the original foliation is singular, is it also the case for the transverse foliation)? Does a transverse manifold always intersect all the leaves?
- How to interpret the leaves since the metric is hyperbolic in the output space (for normal distributions)?

7.2 Possible extensions

- We may visualize the decision boundaries around a point in the same vein as "Robust Learning with Jacobian Regularization".
- Existence of a foliation with respect to the parameters of the model.
- Link between information geometry and Bayesian framework.
- Robust activation (or robust linear model). Maybe already done by Nayebi and Ganguli 2017.
- Natural gradient descent for adversarial attack.
- Use the data leaf as a generative model.
- The data leaf theory might explain **catastrophic forgetting**. When the data leaf re-adapt itself to the new data, it may send all the previous data points to noise leaves.

7.3 To read in machine learning

- Exploring Robust Architectures for Deep Artificial Neural Networks. Exploring Robustness of Neural Networks through Graph Measures. (Rasool et al.)
- $\mathcal{H}$-consistency: On the existence of the adversarial Bayes classifier (Awasthi et al.)
- Certified defense: randomized smoothing, estimation of Lipschitz constants, Theoretically Principled Trade-off between Robustness and Accuracy (Zhang et al. 2019), Globally-robust neural network (Leino et al.), Certified adversarial robustness with additive noise (Li et al.)
- Orthogonal CNN and robustness (is it linked with Parseval networks?).
- Particle filter statistical importance sampling.
- Jacobian regularization: Hoffman et al. [37], and other papers.
- Other attacks and defenses: DeepFool, Defense distillation, Self-supervised learning / semi-supervised learning (they are not the same), Data pruning without annotation.
- What is the **manifold hypothesis** (data points of natural datasets lie "close" to a low-dimensional manifold)? Does it come from manifold learning and dimensionality reduction (see the thesis of Sun Ke 2015)? It seems to be possible to approximate this "data manifold" with generative models (GANs, VAEs). It seems that it is behind the "boundary tilting" explanation and subsequent works. But it doesn't seem to be linked with Fisher information nor Riemannian geometry, how is it possible? It also seems to be linked with Khoury and Hadfield-Menell 2018.
- Visualization of features, deep dream visualization. *This line of research of visualizing the "manifold" of data "preferred" by the model goes back to the deep dream visualizations from google and other efforts to understand the representations learned by deep networks. The authors do not refer to that line of work, and I would be curious to learn how they see their work differing from visualizations of features.*
- VC-dimension and Rademacher complexity, and their links with geometry.
- From [35]: Martens 2020, Sommer & Bronstein 2020, Bergomi et al. 2019, Gabay 1982 (Riemannian gradient descent).
- From [12]: Karakida et al. 2019, Shen et al. 2019
- Application to critical domains in ATM (with certification issues): Sameer Alam (mainly Reinforcement Learning)
- Other keywords for papers: "Manifold", "Fisher", "Geometry", "Lyapunov", "Perceptually Aligned Gradient".

Self-teaching with Hikmat:

- SGD.
- Learning Theory from First Principles (Bach), chapter 3.

7.4 To read in geometry

- Differential Geometry (Tu), Riemannian Foliations (Molino), Geometric Modeling in Probability and Statistics (Calin & Udriste)
- To approximate the exponential map (geodesic retraction), see the *retractions*: polar retraction, Cayley transform, etc. See for example Gao et al. 2018., Monera et al. 2014.
- Skovgaard 1984.
- José C. Principe. Information Theoretic Learning: Renyi's Entropy and Kernel Perspectives.
- A Riemannian Framework for Tensor Computing.

7.5 To read in statistics

- Exponential families & GLM
  Models that are easy to compute and with lots of good properties. But unable to model the complexity of the real world.
- Bayesian statistics & priors: Pratik Chaudhari (reference priors)
- Variational inference & Variational Bayes
- Generative & Discriminative

7.6 Tools

- Python library for information geometry: geostats.

References

[1] K. Tan, J. Wang, and Y. Kantaros, "Targeted Adversarial Attacks against Neural Network Trajectory Predictors," 2022.

[2] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," Feb. 2014.

[3] H. Xu, Y. Ma, H.-C. Liu, D. Deb, H. Liu, J.-L. Tang, and A. K. Jain, "Adversarial Attacks and Defenses in Images, Graphs and Text: A Review," *International Journal of Automation and Computing*, vol. 17, no. 2, pp. 151–178, 2020.

[4] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and Harnessing Adversarial Examples," *International Conference on Learning Representations (ICLR)*, 2015.

[5] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial Machine Learning at Scale," Feb. 2017.

[6] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards Deep Learning Models Resistant to Adversarial Attacks," Sept. 2019.

[7] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "DeepFool: A simple and accurate method to fool deep neural networks," July 2016.

[8] C. Xiao, J.-Y. Zhu, B. Li, W. He, M. Liu, and D. Song, "Spatially Transformed Adversarial Examples," 2018.

[9] 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.

[10] J. Gilmer, L. Metz, F. Faghri, S. S. Schoenholz, M. Raghu, M. Wattenberg, and I. Goodfellow, "Adversarial Spheres," *arXiv:1801.02774 [cs]*, Sept. 2018.

[11] C. Zhao, P. T. Fletcher, M. Yu, Y. Peng, G. Zhang, and C. Shen, "The Adversarial Attack and Detection under the Fisher Information Metric," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 5869–5876, July 2019.

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

[13] A. Ilyas, L. Engstrom, S. Santurkar, B. Tran, D. Tsipras, and A. Ma, "Adversarial Examples are not Bugs, they are Features," in *33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*, (Vancouver, Canada), 2019.

[14] 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.

[15] R. Müller, S. Kornblith, and G. E. Hinton, "When does label smoothing help?," in *Advances in Neural Information Processing Systems*, 2019.

[16] C. Shen, Y. Peng, G. Zhang, and J. Fan, "Defending against adversarial attacks by suppressing the largest eigenvalue of fisher information matrix," 2019.

[17] H. Zhang, Y. Yu, J. Jiao, E. Xing, L. E. Ghaoui, and M. Jordan, "Theoretically principled trade-off between robustness and accuracy," in *Proceedings of the 36th International Conference on Machine Learning*, vol. 97, pp. 7472–7482, 2019.

[18] M. Picot, F. Messina, M. Boudiaf, F. Labeau, I. Ben Ayed, and P. Piantanida, "Adversarial robustness via fisher-rao regularization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[19] J. Cohen, E. Rosenfeld, and Z. Kolter, "Certified Adversarial Robustness via Randomized Smoothing," in *Proceedings of the 36th International Conference on Machine Learning*, pp. 1310–1320, May 2019.

[20] Z. Hao, C. Ying, Y. Dong, H. Su, J. Song, and J. Zhu, "GSmooth: Certified Robustness against Semantic Transformations via Generalized Randomized Smoothing," in *Proceedings of the 39th International Conference on Machine Learning*, pp. 8465–8483, 2022.

[21] D. Dera, G. Rasool, and N. C. Bouaynaya, "Extended variational inference for propagating uncertainty in convolutional neural networks," in *2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP)*, 2019.

[22] D. Dera, G. Rasool, N. C. Bouaynaya, A. Eichen, S. Shanko, J. Cammerata, and S. Arnold, "Bayes-sar net: Robust sar image classification with uncertainty estimation using bayesian convolutional neural network," *2020 IEEE International Radar Conference*, pp. 362–367, 2020.

[23] N. Akhtar, A. Mian, N. Kardan, and M. Shah, "Advances in Adversarial Attacks and Defenses in Computer Vision: A Survey," *IEEE Access*, vol. 9, pp. 155161–155196, 2021.

[24] S. Mahloujifar, D. I. Diochnos, and M. Mahmoody, "The Curse of Concentration in Robust Learning: Evasion and Poisoning Attacks from Concentration of Measure," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 4536–4543, July 2019.

[25] A. Fawzi, H. Fawzi, and O. Fawzi, "Adversarial vulnerability for any classifier," *Advances in Neural Information Processing Systems*, 2018.

[26] A. Shafahi, W. R. Huang, C. Studer, S. Feizi, and T. Goldstein, "Are adversarial examples inevitable?," Feb. 2020.

[27] T. Tanay and L. Griffin, "A Boundary Tilting Persepective on the Phenomenon of Adversarial Examples," Aug. 2016.

[28] A. Shamir, I. Safran, E. Ronen, and O. Dunkelman, "A Simple Explanation for the Existence of Adversarial Examples with Small Hamming Distance," Jan. 2019.

[29] L. Schmidt, S. Santurkar, D. Tsipras, K. Talwar, and A. Ma, "Adversarially Robust Generalization Requires More Data," in *Advances in Neural Information Processing Systems*, p. 13, 2018.

[30] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, and A. Madry, "Robustness May Be at Odds with Accuracy," Sept. 2019.

[31] F. Tramèr, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "The Space of Transferable Adversarial Examples," May 2017.

[32] T. Miyato, S.-i. Maeda, M. Koyama, K. Nakae, and S. Ishii, "Distributional smoothing with virtual adversarial training," 2015.

[33] A. Nayebi and S. Ganguli, "Biologically inspired protection of deep networks from adversarial attacks," 2017.

[34] J. Martin and C. Elster, "Inspecting adversarial examples using the Fisher information," *Neurocomputing*, vol. 382, pp. 80–86, 2020.

[35] L. Grementieri and R. Fioresi, "Model-centric Data Manifold: The Data Through the Eyes of the Model," Apr. 2021.

[36] A. Achille, M. Rovere, and S. Soatto, "Critical Learning periods in Deep Networks," *International Conference on Learning Representations (ICLR)*, p. 14, 2018.

[37] J. Hoffman, D. A. Roberts, and S. Yaida, "Robust learning with jacobian regularization," *ArXiv*, 2018.

[38] A. Shafahi, M. Najibi, M. A. Ghiasi, Z. Xu, J. Dickerson, C. Studer, L. S. Davis, G. Taylor, and T. Goldstein, "Adversarial training for free!," in *Advances in Neural Information Processing Systems*, vol. 32, 2019.