Geometrical perspectives on the robustness of deep neural networks

Geometric data analysis class - Project report

Marius Alonso
Mines Paris PSL
marius.alonso@etu.minesparis.psl.eu

Paul Bonin
Mines Paris PSL
paul.bonin@etu.minesparis.psl.eu

1 INTRODUCTION

Deep learning networks, despite their remarkable capabilities, are highly sensitive to small, imperceptible changes in input data, known as perturbations. This sensitivity can lead to misclassifications and unpredictable behavior in image recognition in real-world (noise-prone) environments.

Understanding how to build robust classifiers is vital for the advancement of safety-critical systems, such as autonomous vehicles or medical diagnosis, in order to minimize the risks associated with adversarial attacks, where maliciously crafted perturbations can manipulate model outputs.

[1], [2], and [5] focus on demonstrating the link between classifiers' robustness and their geometrical characteristics with regard to input spaces, by analyzing the interplay between decision boundaries and vulnerability to perturbations.

2 CONTENT OF THE PAPERS

2.1 Notations

Let $\mathcal{X}$ denote the space of input images. Let $f : \mathbb{R}^d \to \mathbb{R}^L$ denote a L class classifier. The classifier f partitions the space $\mathbb{R}^d$ into classification regions $C_1, \ldots, C_L$ of constant label. Given a datapoint $x_0 \in \mathbb{R}^d$, the estimated label is obtained by $\hat{k}(\boldsymbol{x}_0) = \text{argmax}_k f_k(\boldsymbol{x}_0)$, where $f_k(\boldsymbol{x})$ is the k^{th} component of $f(\boldsymbol{x})$ that corresponds to the k^{th} class. We denote by $\mathcal{R}$ the set of perturbations. For $r \in \mathcal{R}$, we define $T_r : \mathcal{X} \to \mathcal{X}$ to be the perturbation operator by r; i.e., for a data point $x \in \mathcal{X}$, $T_r(x)$ denotes the image x perturbed by r. The pairwise decision boundary of the classifier (between two classes i and j) is defined as the set $\mathcal{B} = \{z : F(z) = 0\}$, where $F_{ij}(z) = f_i(z) - f_j(z)$.

2.2 Perturbations and robustness

2.2.1 Adversarial perturbations. We define the minimal perturbation changing the label of the classifier f, at x, as follows:

$$r^*(x) = \underset{r \in \mathcal{R}}{\text{argmin}} \|r\|_{\mathcal{R}} \text{ subject to } f(T_r(x)) \neq f(x) \tag{1}$$

where $\|.\|_{\mathcal{R}}$ is a metric on $\mathcal{R}$. Note that larger values of $\|r^*(x)\|_{\mathcal{R}}$ indicate a higher robustness of the classifier at x.

The perturbation obtained by solving (1) is often referred to as an adversarial perturbation, as it corresponds to the perturbation that an adversary (having full knowledge of the model) would apply to change the label of the classifier, while causing minimal changes to the original image.

Conference'17, July 2017, Washington, DC, USA
2023. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Deep networks are extremely vulnerable to such perturbations; i.e., small and even imperceptible adversarial perturbations can be computed to fool them with high probability. For example, the typical norm of the average adversarial perturbation required to fool state-of-the-art architectures is 100 times smaller than the typical norm of natural images when using the ℓ_2 norm [1].

2.2.2 Random perturbations. In the random noise regime, data points are perturbed by noise having a random direction in the input space. We measure the pointwise robustness to random noise by setting $\mathcal{R}$ to be a direction sampled uniformly at random from the ℓ_2 unit sphere $\mathbb{S}^{d-1}$ in $\mathcal{X}$ (where d denotes the dimension of $\mathcal{X}$). Equation (1) becomes:

$$r_v^*(x) = \underset{r \in \{\alpha v : \alpha \in \mathbb{R}\}}{\text{argmin}} \|r\|_2 \text{ subject to } f(T_r(x)) \neq f(x) \tag{2}$$

where v is a vector sampled uniformly at random from the unit sphere $\mathbb{S}^{d-1}$. The pointwise robustness is then defined as the ℓ_2 norm of the perturbation, i.e., $\left\|r_v^*(x)\right\|_2$.

State-of-the-art classifiers are much more robust to random noise than to adversarial perturbations, i.e., the norm of the noise $r_v^*(x)$ required to change the label of the classifier can be several orders of magnitudes larger than that of the adversarial perturbation.

2.2.3 Universal perturbations. We then define a universal perturbation v as the minimal perturbation that fools the classifier on a large fraction of the data points sampled from the data distribution μ for a given ℓ_p norm, i.e.,

$$v = \underset{r}{\text{argmin}} \|r\|_p \text{ subject to } \underset{x \sim \mu}{\mathbb{P}}(f(x + r) \neq f(x)) \geq 1 - \epsilon \tag{3}$$

where ϵ controls the fooling rate of the universal perturbation.

By adding this image-agnostic perturbation to a natural image, the label estimated by the deep neural network will be changed with high probability.

Yet, the norm of these perturbations is at least one order of magnitude smaller than the norm of natural images, which makes them hard to detect [1]. This poses a great deal of challenges when it comes to algorithm robustness.

2.3 Curvature

2.3.1 Definition. The curvature profile of the loss function ℓ with respect to the input set (i.e., the curvature profile of $\mathcal{B}$) corresponds to the set of eigenvalues of the Hessian matrix [1]:

$$H = \left(\frac{\partial^2 \ell}{\partial x_i \partial x_j} \right) \in \mathbb{R}^{d \times d} \tag{4}$$

where $x_i, i = 1, \ldots, d$ denote the input pixels (not the network weights). Small eigenvalues (in absolute value) of H indicate a small curvature of the graph of ℓ around x, hence implying that the classifier has a "locally linear" behaviour in the vicinity of x. In contrast, large eigenvalues (in absolute value) imply a high curvature of the loss function in the neighbourhood of image x.

[2] provides a definition of the curvature of the decision boundary, in the case where there are more than two classes. The tangent space of $\mathscr{B}$ at z is noted $\mathcal{T}_z(\mathscr{B})$. The normal curvature $\kappa(z, v)$ along a tangent direction $v \in \mathcal{T}_z(\mathscr{B})$ is defined as the curvature of the planar curve resulting from the cross-section of $\mathscr{B}$ along the two-dimensional normal plane spanning $(\nabla F(z), v)$. Note that for any point on the decision boundary $z \in \mathscr{B}$, the gradient $\nabla F(z)$ is orthogonal to $\mathcal{T}_z(\mathscr{B})$. The curvature along a tangent vector v can be expressed in terms of the Hessian matrix H_F of F :

$$\kappa(z, v) = \frac{v^T H_F(z) v}{\|v\|_2^2 \|\nabla F(z)\|_2} \tag{5}$$

Principal directions correspond to the orthogonal directions in the tangent space maximizing the curvature $\kappa(z, v)$. Specifically, the l-th principal direction v_l (and the corresponding principal curvature κ_l) is obtained by maximizing $\kappa(z, v)$ with the constraint $v_l \perp v_1 \ldots v_{l-1}$.

Alternatively, the principal curvatures correspond to the nonzero eigenvalues of the following matrix

$$\frac{P_F(z) H_F(z) P_F(z)}{\|\nabla F(z)\|_2} \tag{6}$$

Where P_F is the projection operator on the tangent space; i.e., $P = I - \nabla F(z) \nabla F(z)^T / \|\nabla F(z)\|_2^2$. Note that there is an error in the definition of the projection operator in [2].

2.3.2 Link with robustness. The curvature profile of deep networks in highly sparse (i.e., the decision boundaries are almost flat along most directions) but can have a very large curvature along a few directions [1]. Furthermore, the principal curvature profile is asymmetric towards negatively curved directions [2].

[5] explains that adversarial training (i.e. training extra epochs on data that has been applied an adversarial perturbation) induces a significant decrease in the curvature of the decision boundaries of the classifier in the input space. [5] also demonstrates the existence of a strong correlation between small curvature of the decision boundary $\mathscr{B}$ (i.e. local linearity) in all directions in the vicinity of input space data points and classifier robustness.

The high instability of classifiers to adversarial perturbations shows that natural images lie very closely to the classifier's decision boundary [1]. A local geometric description of the decision boundary is provided by the direction of $r^*(x)$ as defined in (1). Indeed, $r^*(x)$ is orthogonal to the decision boundary $\mathscr{B}$ at $x + r^*(x)$ and $\|r^*(x)\|_2 = \text{dist}(x, \mathscr{B})$. Specifically, for a given image x, the perturbed sample $z = x + r^*(x)$ corresponds to the closest point to x on the decision boundary [2].

2.3.3 Universality of boundary curvatures in the vicinity of data points. If the curvatures of the decision boundary in the neighborhood of different data points were uncorrelated, the best universal perturbation would correspond to a random perturbation. This is not the case, as the norm of the random perturbation required to fool 90% of the images is ten times larger than the norm of the universal perturbation required to fool 90% of the images [1]. Therefore, there is a correlations between different regions of the decision boundary, in the vicinity of different natural images.

It is conjectured that the existence of universal perturbations fooling classifiers for most natural images is partly due to the existence of a low-dimensional subspace $\mathcal{S}$ that captures the correlations among different regions of the decision boundary. Specifically, there would be common directions where the decision boundary is positively curved in the vicinity of most natural images. With this geometric perspective, universal perturbations correspond exactly to directions where the decision boundary is positively curved in the vicinity of most natural images [1]. [2] explains that classifiers are invariant to noise in $\mathcal{S}^\perp$, while highly vulnerable to noise in $\mathcal{S}$.

2.4 Topology of classification regions

[2] treats the regions C_i as topological spaces, and studies their path connectness. Formally: given any two data points $x_1, x_2 \in C_i$, they elaborate an algorithm that approximate a continuous curve $\gamma : [0, 1] \to C_i$ exist, such that $\gamma(0) = x_1, \gamma(1) = x_2$. Their conclusion is that the former path always exists between two data points sampled from the same classification region, and that the path connecting those points is approximately a straight one.

The classification regions created by deep neural networks are therefore connected in $\mathbb{R}^d$. Yet, they are not convex bodies. Indeed, the probability that a random convex combination of $k = 2$ images from the same region C_i lies within this classification region is approximately 0.8. However, for larger k, this probability gets much smaller, which implies that classification regions are not convex bodies in $\mathbb{R}^d$ (there are "holes" between the paths that connect the points within the regions C_i).

2.5 Detecting perturbed examples

[2] proposes a method to distinguish between original images, and images perturbed with small adversarial perturbations in order to improve the robustness of classifiers.

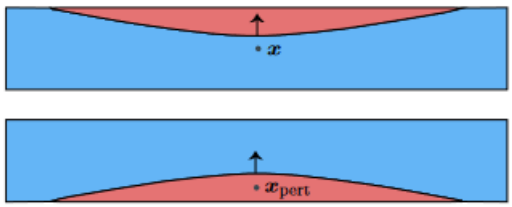

Figure 1: Illustration of point whose curvature of neighboring boundary is negative (above) or positive (below)

For an element z on the decision boundary, denote by $\bar{\kappa}(z) = \frac{1}{d-1} \sum_{i=1}^{d-1} \kappa_i(z)$ the average of the principal curvatures. For points z sampled in the vicinity of natural images, the profile of the principal curvature is asymmetric, leading to a negative average curvature ; i.e., $\bar{\kappa}(z) < 0$. In contrast, if x is now perturbed with an adversarial example (that is, we observe $x_{\text{pert}} = x + r(x)$ instead of x), the average curvature at the vicinity of x_{pert} is instead positive. Based

on this simple idea, [2] achieves classification between perturbed and original images.

3 EXPERIMENT

In order to reproduce some of the results from the above-mentioned papers, we used a LeNet convolutional classifier on the MNIST dataset. It has been chosen for the dimension of the image space $d = 1 \times 28 \times 28$, which is rather small. The goal was to be able to easily compute the eigenvalues of the Hessian matrix.

The same results have been observed when using the Fashion MNIST dataset.

3.1 Computation of the curvature of the decision boundary

3.1.1 Definition and Remarks. Recall the definition of the curvature of decision boundary given in 2.3.1. We rely on this for our assessment.

Note that this definition is valid only for a point z lying precisely on the decision boundary. However, we consider, as [2] did, that a sample x with some label i is close enough to its decision boundaries, so that the curvature of the decision boundary with class j near x can be approximated by the curvature of the surface defined by $\{z : F_{ij}(z) = F_{ij}(x)\}$ in x.

The following figure represents the intersection of $\mathscr{B}$ with the plan defined by the two vectors $\nabla F_{ij}(z)$ and $(z-x) - p_{\vec{n}}(z-x)$.

- z is a point lying on the boundary near x
- y is an adversarial perturbation of x used to find z
- $p_{\vec{n}}(z-x)$ is the projection of vector $(z-x)$ on the (normalized) gradient $\vec{n} = \nabla F_{ij}(z)/||\nabla F_{ij}(z)||$

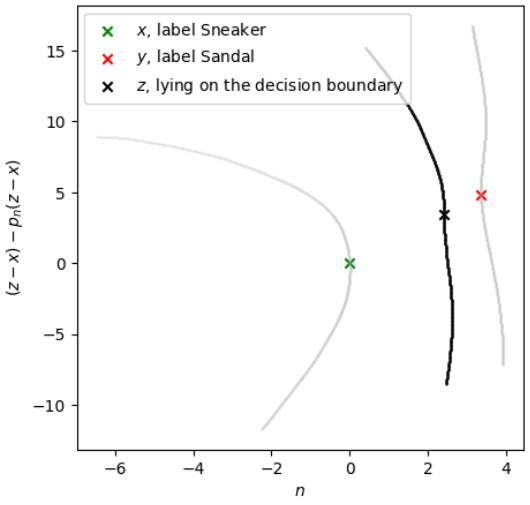

Figure 2: Example of a decision boundary. Note that the axes are graduated in orthonormal coordinates. To find z belonging to the decision boundary near x, an adversarial perturbation y is generated, and then we solve for $t \mid F_{ij}(y+t\cdot(x-y)) = 0$

3.1.2 Methodology. The method used to compute the curvature of the decision boundaries around x makes use of equations (5) and (6). We provide a description of its steps hereafter:

(1) **Compute the gradients** $\nabla F_{ij}(x)$, where i is the label of x and $j \in \{1, \dots, 10\}\backslash\{i\}$.
(2) **Compute the projected orthogonal vectors.** For $k = 1, \dots, N$, compute :
$$v_k = e_k - < e_k, \frac{\nabla F_{ij}(x)}{||\nabla F_{ij}(x)||} > \frac{\nabla F_{ij}(x)}{||\nabla F_{ij}(x)||}$$
(3) **Compute the shifted gradients** $(\nabla F_{ij}(x + h \cdot v_k))_{k=1,\dots,N}$. h is a positive real number, here chosen to be 10^{-1}.
(4) **Extract the coefficients of the matrix** $G(x) = P(x)H(x)P(x)$.
$$\begin{aligned} G(x)_{kl} &= e_k^T (P(x)H(x)P(x))e_l \\ &= (P(x)e_k)^T H(x)(P(x)e_l) \\ &= v_k^T H(x) v_l \\ &\approx v_k^T \frac{\nabla F(x + h \cdot v_l) - \nabla F(x)}{h} \end{aligned}$$
(5) **Retrieve the eigenvalues of** $G(x)/||\nabla F(x)||$. They are the principal curvatures of the decision boundary, as previously explicited with equation (6).

Note that this procedure can be adapted to compute solely the Hessian (and possibly its eigenvalues), by removing the projection step (2), and therefore having $\forall k, v_k = e_k$, and $G(x) = H(x)$.

For step (4), a choice has to be made for the value of h. In [5], in a fairly similar context, they chose a large value, in the magnitude of 10^0. We choose a similar magnitude. Indeed, we are not interested in the local effects, and if h is picked too small, we retrieve a null curvature (with ReLU activation function, the network output is a piecewise affine function of its imput).

3.1.3 Observations. The results obtained are in line with the findings of [1], [2] and [5]: the vast majority of the principal curvatures are close to zero, with a few outliers lying far from zero in absolute value, as illustrated by Figure 3.

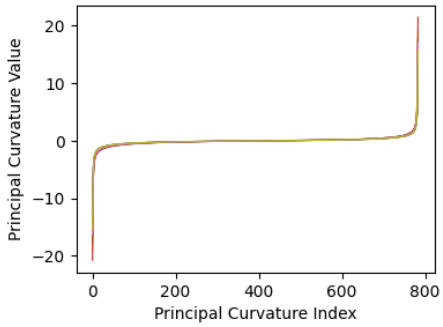

Figure 3: Mean of the Ordered Principal Curvatures of the 9 decision boundaries near 40 images labelled 0

We recall the average curvature of the decision boundaries, thoroughly analyzed in [2], which can simply be computed by taking the mean of the principal curvatures, or by computing the expectancy of the curvature along a random unit vector. The observed average

curvature for our 40-image sample is slightly negative (see Figure 4), which aligns with the observations of [1] and [2].

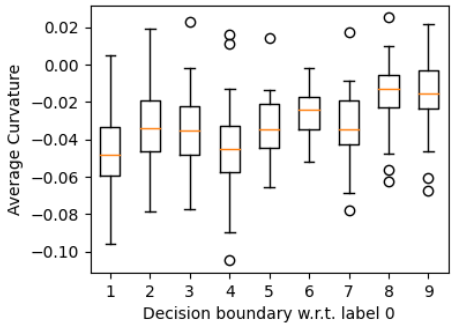

Figure 4: Average Curvatures of the 9 decision boundaries near 40 images labelled 0

3.2 Test against adversarial attacks

To further assess the results found in [2], we perform some adversarial attacks on a sample of images. We use [4] to generate attacks of type PGD, with a maximum tolerated l_∞ error of $\varepsilon = 50/255$. The success rate for this value of ε is around 85%. We also perform a second assessment, keeping the same network and training it on the fashion MNIST dataset. In an attempt to perform attacks of the same *nature*, we equalize the success rate by setting $\varepsilon = 16/255$.

3.2.1 True vs. perturbed classification. [2] claims it is possible to build a classifier of perturbed vs. unperturbed samples based on average curvature. A well defined threshold t can serve to classify them: if the average curvature $\overline{\kappa} = \sum_{j \neq i} \overline{\kappa}_j$ is above this threshold, then the sample is labelled as perturbed, else not.

Here, to compute the $\overline{\kappa}_j$, $j = 1, \ldots, 10$ and $j \neq i$, we rely on another formula given in [3]:

$$\overline{\kappa}_j = \frac{\nabla F_{ij}^T \cdot H \cdot \nabla F_{ij} - \text{trace}(H)||\nabla F_{ij}||^2}{2||\nabla F_{ij}||^3} \tag{7}$$

The classification rule is tested on a sample of 50 pictures, that have undergone an adversarial attack. Among them, we pick 25 at random that will be representative of the true class. Their perturbed version is discarded. The 25 remaining samples are representative of perturbed class, and their true version is discarded.

With an optimal threshold, we obtain a classifier that has an accuracy of $75 - 85\%$ on the 50 picture sample. The ROC curve with varying threshold is displayed in Figure 5. On the fashion MNIST dataset, the results are similar.

3.2.2 Relabelling. In addition, [2] proposes a strategy to find the *original label* of the perturbed sample, i.e. the label given by the neural network to the original picture (before the attack) . Note that here, there is no consideration for the *true label*, understood as the label of the sample in the test dataset.

The proposed *original label* estimate is the label $j^* \neq i$ such that $j^* = \arg\max_j \overline{\kappa}_j$. We evaluate this estimate in our current set-up, and obtain good results. On the Figure 6, you can see, for the 50 perturbed samples, the rank of the average curvature of the boundary

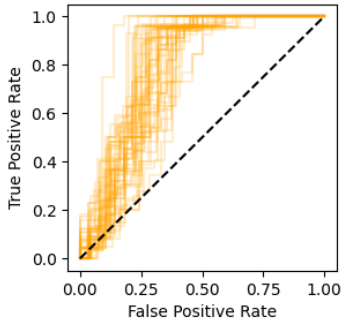

Figure 5: ROC curve of threshold identification of perturbed samples, based on the total average curvature score

with original label. The rank is taken among the sorted list of the 9 decision boundary curvatures $(\overline{\kappa}_j)_{j \in \{0,\ldots,10\} \setminus \{\text{perturbed label}\}}$, in decreasing order. Here, original labels had the highest value about 65% of the time, the second largest value about 20% of the time, etc.

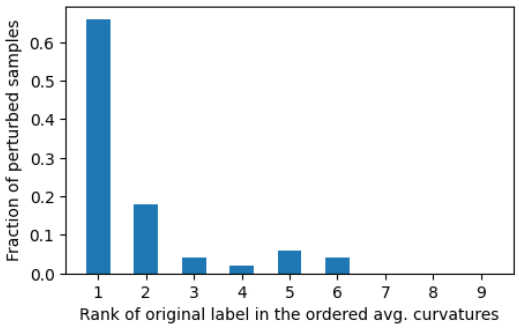

Figure 6: Rank of original label when sorting the sample average curvature $\overline{\kappa}_j$ in descending order

We also evaluate the estimate on the Fashion MNIST dataset and obtain a comparable accuracy of 64%. In both cases, the results are not as significative as [2], and we are not sure about the norm constraints imposed on the attacks of their experimental set-up.

4 CONCLUSION

To sum up, the analysis of decision boundary curvatures affirmed the prevalence of locally linear behavior in the vicinity of data points, along with pronounced curvature along a few directions, which is coherent with literature results. Leveraging LeNet on both MNIST and Fashion MNIST datasets demonstrated a predominance of near-zero principal curvatures in decision boundaries.

Moreover, our assessment on adversarial attacks demonstrated potential in discerning perturbed from original samples, based on average curvature. This could be leveraged to fortify defenses against adversarial perturbations.

Overall, this projects underlines the great importance of adopting a geometric lens in order to understand the resilience of deep neural networks to perturbations.

REFERENCES

[1] Alhussein Fawzi, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. 2017. The Robustness of Deep Networks: A Geometrical Perspective. *IEEE Signal Processing Magazine* 34, 6 (2017), 50–62. https://doi.org/10.1109/MSP.2017.2740965

[2] Alhussein Fawzi, Seyed-Mohsen Moosavi-Dezfooli, Pascal Frossard, and Stefano Soatto. 2018. Empirical Study of the Topology and Geometry of Deep Networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3762–3770. https://doi.org/10.1109/CVPR.2018.00396

[3] Ron Goldman. 2005. Curvature formulas for implicit curves and surfaces. *Computer Aided Geometric Design* 22, 7 (2005), 632–658.

[4] Hoki Kim. 2020. Torchattacks: A pytorch repository for adversarial attacks. *arXiv preprint arXiv:2010.01950* (2020).

[5] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Jonathan Uesato, and Pascal Frossard. 2019. Robustness via Curvature Regularization, and Vice Versa. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 9070–9078. https://doi.org/10.1109/CVPR.2019.00929