Notes

Contents

1 About robustness

Let $F : \mathcal{X} \to \mathcal{Y}$ be a predictor.
No assumptions on $F, \mathcal{X}$ or $\mathcal{Y}$ for now.

1.1 Intuition behind robustness

A predictor is robust if it's prediction is *stable* under *irrelevant* perturbations of its input.
For classification, the *stability* of the prediction corresponds to class change.
Class change is hard to study, because the $0-1$ loss is non-differentiable (?). In practice, we rely on surrogates (is it really necessary?).
Not sure how to define stability for regression.
The *irrelevancy* of the perturbation depends on the choice of a dissimilarity measure on $\mathcal{X}$.
l_p norms are bad dissimilarity measures [1], because they do not reflect the intuitive "indistinguishability".
Among l_p norms, the least worse is l_∞.
Very important. There exist:

1. Perturbations with large l_p norms but still indistinguishable [2]
2. Perturbations with small l_p norms but not indistinguishable [3, 1, 4]

As argued in [3], both accuracy and robustness are defined by reference to an *oracle*[1]. In practice, this oracle is a human.

[1]Here, an oracle is some deterministic or stochastic function that gives a label $y \in \mathcal{Y}$ to any input $x \in \mathcal{X}$. It should not be confused with the Bayes optimal classifier which is necessarily deterministic (if I am correct). It should not be confused with the data distribution P which gives the conditional probability $\mathbb{P}(y|x)$ but not the label y itself.

1. The human labels the dataset, so he is responsible of the accuracy.

2. The human decides if a perturbation is *adversarial* or not (i.e., indistinguishable or not), so he is responsible of the robustness.

But surprisingly, the formal definition of adversarial robustness *does not refer to the oracle*, while the formal definition of accuracy does.
Thus, [3] argue that the *accuracy-robustness trade-off doesn't exist.* If you impose that a predictor should give the same prediction in some l_p ball, then of course, there will be a trade-off between robustness and accuracy! Because there may be an example in this l_p ball with a different label *as given by the oracle*, so you either change the predicted label of this example (sacrifice robustness), or either missclassify this example (sacrifice accuracy).

Wasserstein distance may be a better dissimilarity measure on $\mathcal{X}$ [5, 6, 7]. But it still feels very heuristic.

If you think about it, dissimilarity is an ill-defined notion. For example, two pictures of a cat may be very different in terms of lighting or colors, but they are still very similar because they both contain a cat. They are not indistinguishable, but are they dissimilar? Maybe not. In fact, *indistinguishability is not a relevant concept.* There are small perturbations that are distinguishable but we still want the predictor to be stable i.e., these perturbations are still irrelevant!

Now, what if we are classifying the trees in the background and not the cats? Maybe the images are dissimilar because the trees are different. Finding a good or a "perfect" dissimilarity measure is already a learning task. It amounts to finding the *relevant features* to discriminate two inputs. *Dissimilarity measures are task-dependent.* So maybe, **the definition of robustness is task-dependent.**

1.2 About verification of model stability

Following the literature, I used to talk about "certified robustness". Now, I think it is a bad terminology. First, the word "robustness" is too vague. There are two types of robustness:

1. Robustness in adverse conditions.

2. Model stability.

We only focus on model stability. Then, the word "certification" is misleading, because it can be confused with the certification process in safety-critical domain such as aviation, which is a completely different beast (for which model stability is a one consideration among many others). On the other hand, the word "verification" is already used in classical programming for exactly the same purposes as ours. Thus, I will now talk about "verification of model stability".

In my current understanding, there are three types of verification methods for machine learning model stability:

1. Optimization-based. This includes all exact methods. In particular, it has been shown [8] that exact certification of a ReLU network is NP-hard. There are also methods using tools from optimization such as convex relaxation. I don't know the precise functioning of these methods for now, but every paper I read argues that none of them scale to even medium-sized networks. After reading some reviews, it seems that these methods (which can be either exact or approximate) are extensions of an old line of works called *formal methods.* It includes techniques such as *abstract representation*, *interval bounding*, or *reachability property.* Some methods relies on "classical" optimization: Mixed-Integer Linear Programming, and Satisfiability Modulo Theory (SMT) (including Boolean Satisfiability Problem (SAT)). These are quite mathematical topics, with a large literature ranging over several decades. Honestly, I don't have the courage nor the time to study all of this. Besides the scaling issue, this is the main reason why I will focus on other approaches.

2. Lipschitz-based. One approach is to estimate or bound the local or global Lipschitz constant. Our "partial isometry regularization" fall into this category. Note that it can also be based on methods from optimization, so the name "optimization-based" is a little bit misleading. The Lipschitz constant is often upper-bounded by the product of the spectral norm of each layer. Another more promising approach is to train Lipschitz network, i.e., networks with a fixed Lipschitz constant "by-design". It

relies on orthogonal constraints, since orthogonal transformations have Lipschitz constant equal to 1, because their spectral norm (highest singular value) is equal to 1 (in fact, all singular values of an orthogonal transformation are equal to 1).

3. Statistics-based. There are several methods but the most well-known is randomized smoothing. It is claimed to be the only certification method that is scalable, even if some limits have been exposed. It's main drawback is that it is a probabilistic method, thus the certification is not ... certified, and it requires Monte Carlo sampling with a number of samples that grows quickly.

My intuition is that all these verification methods are wrong. Because they separate the verification from the training. Neural networks as they are currently trained have a weird notion of simplicity. They are unable to learn the identity map! See also [9]: neural networks are unable to learn tasks that seem trivial. They always create needlessly complicated decision boundaries, they don't understand the tasks. I feel this is the main reason behind the lack of robustness, and also behind a lot of other problems that seem unrelated: out-of-distribution examples, unbalanced datasets, catastrophic forgetting, distribution shift ...

In the current verification methods, we take the training process and the trained network for granted. Then, we look for tricks to certify them: what are their Lipschitz constant, in what region are they constant etc. Robustification is eventually done *on top of classical training*: data augmentation with adversarial or noisy examples (i.e., adversarial training), adding another term to the cross entropy (and artificially creating a trade-off between robustness and accuracy!). While we know that accurate and robust models exist [3]! Why aren't we converging to them (maybe a beginning of answer in [3])? We must change the training process to train networks that actually understand what they are doing. Then this will *imply* robustness, generalization etc.

I feel like this line of argumentation has already been addressed. Yet, it is never discussed in recent papers (except a little bit in [9]). Should I read older papers, older books? Books on statistical learning theory (like Francis Bach's last book)?

I remember reading somewhere people saying: "We don't care about creating "intelligence", or "understanding". We see machine learning and neural networks as useful technologies (are they?), and we are only interested in their applications. We must separate the research on intelligence (and even more the research about the brain) and practical machine learning research". Somehow, I agree that we should separate the buzz from the practice. But I strongly disagree with the idea that we could achieve meaningful applications without solving the scientific problem of "intelligence". We can't deploy machine learning in critical applications without understanding robustness, and we can't understand robustness without understanding intelligence. And I am not even talking about data poisoning and privacy issues that may be far more concerning (I don't know the topic enough): they may transform machine learning applications into our worst nightmare. Once again, we can't understand poisoning issues without understanding intelligence. Pursuing fast applications to earn easy money can only lead to moral disasters.

1.3 The two Problems of model stability

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

The two Problems of model stability are:

1. How to train models that "understand what they are doing", and how to verify that they understand?

2. How to choose the dissimilarity measure?

Addendum. I think now that the second problem is a consequence of the first one. Choosing the "perfect" dissimilarity measure is equivalent to solving the task perfectly, i.e., understanding the task. Moreover, I have more and more doubts about ensuring the "model stability" by enforcing the model to be constant in a ball of a certain dissimilarity measure (around points that are not "too close" to the true decision boundary). This is because of:

- Invariance attacks vs stability attacks. Maybe, invariance attacks are just equivalent to drop of accuracy, thus this would be equivalent to the "accuracy-robustness trade-off". Anyway, having a model that is too stable is not desirable. Moreover, I think that formulating the problem as a "trade-off" between stability (or robustness) and accuracy is misleading. The real problem is *how can we design models that understand the task?*
- This is a local criterion. Different points may have different robust radius. Some points must have zero or almost zero robust radius (in order to be able to change the class). What metric should we use to quantify this global robustness?
- This may simply be impossible, because almost any point of "natural" high-dimensional dataset may be close to a true decision boundary.

1.4 More discussions on the choice of the dissimilarity measure

I already argued that the "right" dissimilarity measure is task-dependent. In [4], the authors show that finding the right dissimilarity measure is equivalent to solving the task (which is obvious, I don't know why I didn't figured this out myself). Thus, any other dissimilarity measure used to "certify" the robustness of a network will create either sensitivity attack (beyond the certified radius) or invariance attack, and often both.
Nidhal says that invariance attacks are similar to an accuracy drop which is a well-known phenomenon (so-called "robustness-accuracy trade-off"). Moreover, she says that contrary to sensitivity attacks, there is no real-world use-case for invariance attacks. I don't know how to respond to this, but I feel that something is wrong.

What do I mean by "solving a task"? Here is the classical framework, as far as I understand. We assume that there exists an **oracle** $h : \mathcal{X} \to \mathcal{Y}$. Or at least, there exists a probability kernel $\kappa(y|x)$. A task is ϵ-solved by a model f if $\|h - f\|_\infty \leq \epsilon$.
Now, I propose a definition of adversarial attacks. Assume that we have some "neighborhood" $\eta(x)$. We say that a point x' is an adversarial attack with respect to x if:

1. $x' \in \eta(x)$,
2. $f(x) = h(x)$,
3. $f(x') \neq h(x')$.

This definition include both sensitivity and invariance attacks.
Here is another definition including only sensitivity attacks:

1. $x' \in \eta(x)$,
2. $f(x) = h(x)$,
3. $h(x') = h(x)$,
4. $f(x') \neq f(x)$.

Of course, this definition is included into the former. In the following, I will rely on the first definition.

We said that, for any $\eta(x)$, there will exist points x' that are adversarial attacks for x. Unless $\eta(x)$ contains only points that have the class $h(x)$ according to h. The classical criteria for choosing $\eta(x)$ is that the perturbed point x' should be "almost indistinguishable" from x according to humans, which is an ill-defined criteria. If we add a big and clearly visible perturbation to x such that $f(x') \neq h(x')$ but the class $h(x)$ is unchanged, we will call it a classification error. Why should we not call this an adversarial attack? Because, we want to stick to the scenario where an adversary slightly perturb your input. However, this scenario almost never happens in real-life. And even if an adversary could modify the input of a model, why should he limit himself to small perturbations? Since there is probably no mechanism to separate normal from perturbed images, there is no constraint for the adversary.

Thus, robustness is in fact very close to generalization. Could we imagine a model with very good generalization but poor robustness? Yes, because generalization is measured with the expectation over the true distribution $p(x, y) = \kappa(y|x)q(x)$. If a model is good for likely x (large $q(x)$) but bad for unlikely x (small $q(x)$), it will have good generalization but poor robustness. To learn a robust model, **we should increase the probability of unlikely** x, i.e., the data points that are "closer to adversarial examples". In the training data, unlikely inputs correspond to "isolated" data points. During training, these isolated data points should be given more weight.
The general idea is that the loss function is the key. We should replace cross-entropy by a loss function that converges to an estimator with better properties than MLE. Miscellaneous remarks:

- We want to estimate $p(x, y; \theta) = p(y|x; \theta)q(x)$. What is the impact/bias of $q(x)$ on the learned parameter θ?
- Lipschitz neural network is a good architecture: optimization (consistency of cross-entropy, no vanishing/exploding gradients), generalization (finite VC dimension, PAC-learnability, consistency of 0-1 loss, i.e., no overfitting), robustness (margin-based certificates), but it still uses L_p norms! How LNN behave w.r.t. invariance, semantic, or physical attacks?

Some ideas for applying information geometry to machine learning:

- MLE seems to be the entropy-minimizing estimator. Maybe, we should add some kind of "regularization" to the entropy. It may depend on the function $C(x)$ in the exponential family (if it makes any sense to choose a nonzero $C(x)$ for categorical distributions). The loss function becomes $H(\theta) - \mathbb{E}_\theta[C(x)]$. Or maybe, we should minimize another divergence (rather than the KL divergence). Choosing another α and its canonical divergence $D^{(\alpha)}$? Another f-divergence?
- Look for a loss function converging to an estimator with better high-order efficiency? For example, the bias-corrected MLE for second-order efficiency. Something else for third-order efficiency (since MLE is not third-order efficient).
- Another vague idea: some kind of "Bayesian duality" between $p(\theta|x)$ and $p(x|\theta)$ where we see θ as a random variable (similarly to VDP). This is discussed in a paper from Amari, but not clear.
- Once we have the "correct" loss function, there is still the question of optimization. The loss function should as far as possible be trainable with SGD. The higher-order information should be carefully limited and estimated to limit the computation cost (some kind of fast natural gradient).
- Link between inductive bias/priors and information geometry.
  Why bias-variance trade-off should in theory imply that NN are impossible to train?
  Why is it possible to train NN? Implicit regularization: NN favor "simple" solutions that generalize well.
  Is it linked with architecture (then what about LNN implicit regularization?) and/or to optimization (then what about natural gradient?).
  What is the link between training NN and MLE? What properties of MLE are incompatible with robustness?

For the moment, we assume that the data are iid and "perfect". Once, we have a working method, we will think about: distribution shift (i.e., no identically distributed), dependent data, unbalanced data, poisoning/label errors, noisy data, few data.

1.5 Some facts concerning exponential families and KL divergence

Why it matters that KL divergence is not symmetric?

- $\min_{q \in M} D(p||q)$ is equivalent to finding the MLE, where p is the empirical distribution of the data, and q belongs to some parameterized family M.
- $k = \min_{p \in M} D(p||q)$ is linked to Large Deviation Theory. If p belongs to some subset M, and q is the true distribution of the data, then the probability of obtaining a sample whose empirical distribution is in M is asymptotically equal to $\mathbb{P}(M) = \exp(-k)$.

- The symmetrized KL divergence has a geometrical interpretation. Let p and q be two distributions. We can join p and q by two curves: $\gamma_m(t) = (1-t)p + tq$ (mixture family) and $\gamma_e(t) = \exp((1-t)\log(p) + t\log(q) - \psi(t))$ (exponential family). Both curves have the same length (using Fisher metric) which is equal to $\frac{D(p||q)+D(q||p)}{2}$.
- Another intuition. Imagine we want to fit a bimodal distribution q (e.g., mixture of two gaussians) with a unimodal distribution p (e.g., a gaussian). If we minimize $D(q||p)$ then p will spread across both modes of q, with high variance and a mean in-between the two modes. If we minimize $D(p||q)$, then p will closely fit the highest mode of q while ignoring the other.

I don't know how all of this is related.

More intuitions about KL divergence Imagine that we have a given ellipsis, and we want to find the circle that is the "closest" (or the "more similar") to the ellipsis. There are two solutions:

- Every part of the circle is also a part of the ellipsis. Then, the circle is included into the ellipsis. This corresponds to minimizing KL[circle||ellipsis]. Everything that is not in the circle is neglected, but when we are inside the circle, we must be inside the ellipsis. Here, the "variance" is under-estimated (the circle is smaller than the ellipsis).
- Every part of the ellipsis is also a part of the circle. Then, the ellipsis is included into the circle. This corresponds to minimizing KL[ellipsis||circle]. Everything that is not in the ellipsis is neglected, but when we are inside the ellipsis, we must be inside the circle. Here, the "variance" is over-estimated (the circle is bigger than the ellipsis).

Now, let's come back at the problem of fitting data sampled from a bimodal distribution (the true distribution) using a unimodal distribution (the model). Once again, there are two solutions.

- We want the model to be representative of *all* the data. In other words, the data must be "inside" the model (as much as possible). Yet another way to say it is that we *integrate* with respect to the data. This corresponds to minimizing KL[data||model], which is equivalent to Maximum Likelihood Estimation.
- We want the model to *generate* likely data with large probability. We don't care if some part of the data are never generated. Here, the model must be "inside" the data, thus we *integrate* with respect to the model. This corresponds to minimizing KL[model||data], which is equivalent to obtaining (the exponent of) the probability of obtaining data that seems to come from a model when they really come from the true distribution. This is Large Deviation theory. In other words, we are looking for the model (among the allowed family of unimodal models) that is most likely to generate data sampled from the true bimodal distribution. In practice, this will lead to *mode collapse*, i.e., the model will focus on one mode and neglect the other. This is a problem for generative model. It seems to arise when the model family is not complex enough to capture the entire data distribution.

Why are exponential families so important/natural? There are at least 4 characterizations/interpretations of exponential families:

- Existence of a sufficient statistic of fixed dimension.
- Existence of an unbiased first-order efficient estimator for finite data.
- Exponential families are solution to an optimization problem. Maximize the entropy subject to "moments constraints" $\mathbb{E}[F_i(x)] = \lambda_i$ for some functions F_i and real numbers λ_i. The natural parameters θ can be seen as Lagrange multipliers of this optimization problem. This is probably the more fundamental interpretation.
- Exponential families are dually flat spaces. Dually flat spaces are the information geometry equivalent of Euclidean spaces, where the Euclidean distance is replaced by the KL divergence.

Other settings for statistical inference.

- If the data are (x, z) but only x is observed. How can we estimate the parameters $p_\theta(x, z)$? Answer: we can use the EM (expectation-maximization) algorithm. EM algorithm is a special case of the *em* (exponential-mixture) algorithm in information geometry.

- If the parameters are (u, v_i) where u is fixed but v_i varies with each observation x_i. How can we estimate u? This is the Neyman-Scott problem. In this setting, MLE looses all its good properties. There is also an interpretation using information geometry but I haven't studied it.

Small remark concerning Lagrange multipliers. The problem is min $f(x)$ s.t. $g(x) = c$, where $f : \mathbb{R}^n \to \mathbb{R}$ is C^1 and $g : \mathbb{R}^n \to \mathbb{R}^d$ is C^1 with $d < n$. The method of Lagrange multipliers gives a *first-order necessary* condition for a local solution of the problem. Let $M_c = \{x \in \mathbb{R}^n : g(x) = c\}$ and $S_\lambda = \{x \in \mathbb{R}^n : f(x) = \lambda\}$. If the Jacobian matrix of g is always full rank, then M_c is a $(n-d)$-dimensional submanifold. If the gradient of f is always nonzero, then S_λ is a $(n-1)$-submanifold. The geometrical/analytical intuition is the following. If x^* is a solution of the problem, then $T_{x^*} M_c \subseteq T_{x^*} S_{f(x^*)}$ which is equivalent to $N_{x^*} S_{f(x^*)} \subseteq N_{x^*} M_c$ (inclusion of normal spaces). In other words, M_c is tangent to $S_{f(x^*)}$. This is because if this condition is not met, then $S_{f(x^*)}$ crosses M_c. Thus, by moving slightly from x^*, we can find another point $x \in M_c$ which belongs to some S_λ with $\lambda \leq f(x^*)$. The normal space $N_{x^*} M_c$ is spanned by $\nabla g_1(x^*), \dots, \nabla g_d(x^*)$, while $N_{x^*} S_{f(x^*)}$ is spanned by $\nabla f(x^*)$. Thus, the condition $N_{x^*} S_{f(x^*)} \subseteq N_{x^*} M_c$ is equivalent to the existence of $\theta^1, \dots, \theta^d$ such that $\nabla f(x^*) = \sum_{i=1}^d \theta^i \nabla g_i(x^*)$.
The KKT conditions is a generalization of the Lagrange multipliers for conditions $h(x) \leq c$.
I don't know where is the Legendre duality/Hamiltonian here. The dual problem seems to be using θ as variable instead of x, but there is no η. The KKT conditions are also called the *saddle-point theorem* because the solution (x^*, θ^*) is a minimizer along x and a maximizer along θ, thus a saddle point for $\mathcal{L}(x, \theta) = f(x) - \theta^i g_i(x)$.
It seems that I can imagine a counter-example to Lagrange multipliers. In $\mathbb{R}^3$, let $f(x, y, z) = z^2$ and $g(x, y, z) = (x, y)$ (and $c = (0, 0)$). The admissible set is the z-axis. The minimum of f is achieved at the origin ... No in fact it works because the gradient of f and of g_1, g_2 are all equal to 0 at the origin.

1.6 Robustness and prior selection

1.6.1 Introduction and discussion

I believe that adversarial vulnerability is due to an "imperfect notion of simplicity".
Hypothesis 1. In high dimension and for any "natural" dataset, for almost any point, the true boundary is very close to the chosen point. Closeness is measured using "natural distances" (L_p norms, Wasserstein etc.) in a "natural" coordinate system (e.g., pixel-coordinates for images). This is a consequence of the *curse of dimensionality*.
The first cause of adversarial vulnerability is *overfitting*, which can be seen even in low dimension. However, if there is no overfitting, adversarial vulnerability is still possible in high dimension. Why? Because, if Hypothesis 1 is true, then any discrepancy in the generalization between the model and the true distribution will lead to regions of sensitivity examples and regions of invariance examples. It is possible to solve this issue by sampling more data in the problematic regions.
Hypothesis 2. The problematic regions corresponds to very unlikely regions, i.e., $q(x) \approx 0$.
Thus, it is required to sample *a lot* more data in order to have sufficient data points in the problematic regions. Another solution is to augment the training set by other means such as adversarial training, semantic perturbations (i.e., rotation, brightness, reflection), or noise. Each of these solutions will change $q(x)$ in order to sample from a problematic region. Both adversarial training and semantic perturbations only sample from a small subset of all problematic regions, depending on the choice of attacks and perturbations. Noise can sample everywhere, but without focusing on very unlikely problematic regions, which is very inefficient. The human brain is able to learn from few data, so there must exist another solution.
Hypothesis 3. Solving adversarial vulnerability is possible using Bayesian machine learning *if we use the "correct" prior*.
When a model learns in high dimension, the training data are necessarily "sparse". The model has many choices about how to choose the decision boundary in regions where $q(x) \approx 0$ (i.e., regions with almost no training points). What is the "correct" generalization? It depends on the chosen notion of simplicity, which

is itself a property of the data. The notion of simplicity corresponds to the regularization, or equivalently, the prior in Bayesian inference. There are two questions concerning the "correct prior":
Question 1. Is there an optimal prior for any "natural" data, or is the prior dependent on the type of data?
Question 2. How can we find the "correct" prior?
The "no-free lunch paradigm" might suggest that the optimal prior depends on the data. Concerning the human brain, it seems that the brain is organized into modules which are specialized with a specific task/data type. Each module might have its own prior. This prior was selected through evolution, thus it was "learned" using far more data than a single brain will ever see in its life. Is evolution the most computationally efficient way to learn prior? If yes, it would mean that building intelligent machines will require learning the correct prior *for each task*, which will require large quantity of data. Each new task will require a huge amount of data before being able to build models with the correct prior. This means that only big institutions or companies will be able to build intelligent machines.
Another possibility is that the optimal prior is some sort of "law of nature", such that every data in nature obey the same prior, or the same family of priors.

1.6.2 Geometry of conjugate priors

The conjugate prior of an exponential family is itself an exponential family. The canonical parameters of the prior are the α_i (associated to θ^i) and β (associated to $\psi(\theta)$ which becomes part of the sufficient statistic). The prior consists in adding β pseudo-observations whose empirical distribution is an exponential family with parameters α/β. Then, the MAP is the Bregman median:

$$\theta^{MAP} = \arg\min_{\theta} \sum_{t=1}^{n} D(\theta_t||\theta) + \sum_{t=1}^{\beta} D(\alpha/\beta||\theta).$$

Small remark concerning median and centroid. Let x_i, $i = 1, \ldots, n$ be points of a metric space with metric d. The *centroid*, or center of mass (or mean?) of the x_i is:

$$x_c = \arg\min_{x} \sum_{i=1}^{n} \left(d(x, x_i)\right)^2.$$

With the Euclidean distance, the first-order condition is:

$$\sum_{i=1}^{n} (x_i - x_c) = 0.$$

Thus:

$$x_c = \frac{1}{n} \sum_{i=1}^{n} x_i.$$

The physical interpretation of centroid is that each point x_i pulls x_c towards x_i with **a force proportional to the distance** between x_c and x_i. The centroid is achieved when all the forces are at an equilibrium.
The *median* of the x_i is:

$$x_m = \arg\min_{x} \sum_{i=1}^{n} d(x, x_i).$$

With Euclidean distance, the first-order condition is:

$$\sum_{i=1}^{n} \frac{x_i - x_m}{\|x_i - x_m\|} = 0.$$

There is no closed-form formula for the median.
The physicial interpretation of the median is that each point x_i pulls x_c towards x_i with **a fixed force** (every point has the same force, unless we use weights). The median is achieved when all the forces are at an equilibrium.

An *average* is a general terminology such that the centroid and the median are special cases. It also includes the cases when the $d(x, x_i)$ are weighted.
Since a divergence (such as a Bregman divergence) can be interpreted as an "almost" squared distance, then we should talk about Bregman centroid rather than Bregman median (?) What is the physical interpretation of the Bregman median? See "On the strategy proofness of the geometric median" (El-Mhamdi et al.) for properties of the median.

2 Geometry of Logistic Regression

3 Evaluation of model stability

How to evaluate the robustness of a machine learning model?
There are two different "levels" of robustness:

1. The robustness at a given test example.
2. The "robustness of the model" which is an ill-defined concept. It is generally defined as the average robustness across the test set.

There are two different but related questions:

1. Given a budget (i.e., perturbation size), is there an attack that can fool the model at a given test example? What proportion of test examples are fooled at this budget?
2. What is the smallest perturbation that can fool the model at a given test example?

At the model level, both questions can be answered in one graph with the budget in x-axis and the average number of test examples that can be fooled on the y-axis. We call such graph the robust graph.

Given an input metric (e.g., some L_p norm), empirical evaluation consists in:

- Choosing a specific attack, in general with a specific budget.
- Evaluate the proportion of examples from a test set that are well-classified under the attack.
- Repeat with different budgets.

Some remarks:

- For a given test example, a successfully attack only provides an upper-bound on the smallest perturbation. An unsuccessful attack provides no information: the smallest perturbation may be larger or smaller.
- Since the evaluation is done over a test set, we have no guarantees for point outside the test set. It may be possible to obtain probability bounds.

Given a test example, verification consists in providing a lower-bound on the smallest perturbation. Verification is done for a given test example, but we have no guarantee on the verified radius on a "global level" (i.e., outside the test set). There may still exist areas where the verified radii are trivially small. This leads us to *confidence attacks*, i.e., reducing the confidence of the model without changing the predicted class.

A model must have areas of low confidence (close to the decision boundary). These areas can be tagged with an additional label $\perp$, meaning that the model is too unconfident to make a prediction. The question becomes: for a given confidence margin ϵ, how can we minimize the volume of these "unconfidence regions"?

A few words about RobustBench and AutoAttack. The authors have decided to exclude several defenses from their model zoo:

- Randomized defenses, thus including randomized smoothing as well as my ideas of random predictions. They claim that other evaluation techniques are more suited to evaluate randomized defenses, such as EoT (see the "obfuscated gradients" paper).

- Obfuscated gradients methods (see the aforementioned paper). This includes methods where the input-output gradient is zero, where black-box attacks are better than white-box attacks, or where only the probabilities are available (not the logits). Such defenses make gradient-based attacks less effective, but do not remove the adversarial examples.

- Dynamic defenses, i.e., defenses using optimization at inference time. I guess that such defense could find an adversarial example and move the input point away from it, which once again do not remove the adversarial examples but may fool AutoAttack.

The authors mentioned "adaptive attacks" which seem to be general guidelines to create attacks that can effectively evaluate robustness (see Tramèr et al., 2018).

I find a paper trying to criticize AutoAttack. According to the abstract, their main critic is that AutoAttack is too strong and generates perturbations that can easily be detected by adversarial detection methods. It seems to be particularly true for high-resolution images. Moreover, they claim that there exists attacks which produce smaller perturbations with similar fooling ratio.

4 Partial isometry regularization

4.1 Problems

I list the issue of the partial isometry regularization:

1. It is very dependent on the choice of the hyperparameter η, but also and more importantly on the **initialization**. It feels like minimizing the cross entropy and minimizing the regularizer are completely antagonist objectives. SGD will either converge to low regularization but very low accuracy (i.e., random guess) or to high accuracy but high regularization (as a baseline model). There is a a very delicate equilibrium that you can randomly achieve by training with several initializations, where the robustness is high but the accuracy is not too bad. Note that there is still a significant decrease in clean accuracy.

2. The method is still worse than adversarial training against any metric (except Gaussian noise).

3. The method is worse than Parseval by a small margin. But Parseval has the advantage of not being dependent on the initialization or on any hyperparameter!

4. Somehow, I feel like Parseval is very similar in its rationale, but more sophisticated, since it enforces an isometry *at each layer*. Is it possible to design a regularization term for isometry that takes into account each layer (as suggested by Greg). The other advantage of Parseval is that it structurally impose a small Lipschitz constant for any input, while the ISO method only imposes conditions on the training examples.

5. Why is the Jacobian regularization not working? More generally, why enforcing a Lipschitz constant close to 1 works, but enforcing a Lipschitz constant *smaller than 1* does *not* work? This should be a less constraining condition with the same robustness effect!

6. I have more and more doubt about the usefulness of the Fisher information metric. I am still unable to explain precisely what benefits are brought by choosing the FIM on the output space. But if we remove the FIM and use for example the Euclidean metric on the output space, what difference is there between ISO and any other Lipschitz-based method, like Parseval?

7. ISO is not certifiable! I feel more and more that non-certifiable defenses have been proved to be useless.

The real challenge for machine learning robustness is to design a defense that is:

(a) Certifiable.

(b) Computationally efficient, i.e., scalable to high-dimensional datasets, and large networks.

(c) That certify against any "meaningful" attack, not only l_p but also "semantic attacks" (or spatial attacks, unrestricted attacks, Wasserstein attacks etc.).

(d) That certify with "large enough" radii.

(e) What about noise, out-of-distribution examples, distribution shift[2]? Unfortunately, I don't know very much about these topics, but I feel that the field of machine learning has been divided into all these big problems, while they are all closely related. It is impossible to understand and "solve" robustness alone.

Or to prove that such method cannot exist.

References

[1] M. Sharif, L. Bauer, and M. K. Reiter, "On the suitability of lp-norms for creating and preventing adversarial examples," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018.

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

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

[4] 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.

[5] E. Wong, F. R. Schmidt, and J. Z. Kolter, "Wasserstein adversarial examples via projected sinkhorn iterations," *CoRR*, vol. abs/1902.07906, 2019.

[6] A. Levine and S. Feizi, "Wasserstein Smoothing: Certified Robustness against Wasserstein Adversarial Attacks," 2019.

[7] T. A. Bui, T. Le, Q. Tran, H. Zhao, and D. Phung, "A Unified Wasserstein Distributional Robustness Framework for Adversarial Training," 2022.

[8] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer, "Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks," in *Computer Aided Verification*, 2017.

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

[2] Maybe, we can also include: unbalanced data, unlabeled data, continual learning, few-shot learning ...