MODEL-CENTRIC DATA MANIFOLD: THE DATA THROUGH THE EYES OF THE MODEL

Anonymous authors
Paper under double-blind review

ABSTRACT

We discover that deep ReLU neural network classifiers can see a low-dimensional Riemannian manifold structure on data. Such structure comes via the *local data matrix*, a variation of the Fisher information matrix, where the role of the model parameters is taken by the data variables. We obtain a foliation of the data domain and we show that the dataset on which the model is trained lies on a leaf, the *data leaf*, whose dimension is bounded by the number of classification labels. We validate our results with some experiments with the MNIST dataset: paths on the data leaf connect valid images, while other leaves cover noisy images.

1 INTRODUCTION

In machine learning, models are categorized as discriminative models or generative models. From its inception, deep learning has focused on classification and discriminative models (Krizhevsky et al., 2012; Hinton et al., 2012; Collobert et al., 2011). Only later, generative models based on neural networks have been proposed (Kingma & Welling, 2014; Goodfellow et al., 2014; Van den Oord et al., 2016; Kingma & Dhariwal, 2018). Both kinds of models give us information about the data and the similarity between examples. In particular, generative models introduce a geometric structure on generated data. Such models provide an explicit function that converts a random low-dimensional vector to an example sampled from a probability distribution approximating the one of the training dataset. As proved by Arjovsky & Bottou (2017), generated data lay on a countable union of manifolds. This fact supports the human intuition that data (or a subset of them) have a low-dimensional manifold structure, but in generative models the dimension of such a manifold is usually an hyper-parameter fixed by the experimenter. A recent algorithm by Peebles et al. (2020) provides a way to find an approximation of the number of dimensions of the data manifold, deactivating irrelevant dimensions in a GAN. Here we try to understand if a discriminative model conveys similar information about the data.

In the supervised learning scenario, a model is trained on a labeled dataset to identify the correct label of every input data. Such a trained model has to discard most of the variations of a single example to retain only the features relevant to solve the classification task. A neural network classifier builds a hierarchy of representations that encodes increasingly complex features of the data (Olah et al., 2017). Through the representation function, a distance (e.g. euclidean or cosine) on the representation space of a layer endows input data with a distance. This pyramid of distances on examples is increasingly class-aware: the deeper is the layer, the better the metric reflects the similarity of data according to the task at hand. This observation suggests that the model is implicitly organizing the data according to a suitable structure.

Unfortunately, these intermediate representations and metrics are insufficient to understand the geometric structure of data. First of all, representation functions are not invertible, so we cannot recover the original example from its intermediate representation. Therefore, we cannot interpolate between data points to visualize the variations that affect model predictions. Moreover, the domain of representation functions is the entire data domain $\mathbb{R}^n$. This domain is mostly composed of meaningless noise and data occupy only a thin region inside of it. So, even if representation functions provide us a distance, those metrics are incapable of distinguishing between meaningful data and noise.

We find out that a classification neural network model tells us much more about data. A neural network implicitly identifies a low-dimensional submanifold of the data domain that contains real data. We can prove that if the activation function is piecewise-linear (e.g. ReLU), the neural network

decomposes the data domain $\mathbb{R}^n$ as the disjoint union of submanifolds, the leaves of a foliation. The dimension of every leaf, in particular the dimension of the *data leaf*, is bounded by the number of classes of our classification model. This result stems from the study of the properties of a variant of the Fisher Information matrix, the *local data matrix*.

The low-dimensional data manifold defined by the model is not the anthropocentric data manifold composed of data meaningful for a human observer. The model focuses its attention on discriminating points on this manifold, while it interprets the directions orthogonal to the manifold as noise. The model does not know the anthropocentric data manifold, but it builds its internal geometric structure to represent data to effectively classify them. Along noise directions, i.e. directions moving away from the data manifold, model predictions are often very confident, in general more confident than a human would be.

We can visualize the model-centric manifold using a Riemannian gradient descent algorithm. We report some experiments performed on the MNIST dataset. We compute paths joining data points in the training or test set and show the intermediate points found on the manifold. These experiments give us empirical evidence that all valid data stay on the same leaf of the foliation.

Experiments reveal that the model-centric manifold appears different from the anthropocentric one: in fact, the model-centric manifold comprises images that do not belong to a precise class. The model needs those transition points to connect points with different labels. At the same time, it understands that such transition points represent an ambiguous digit: on such points, the model assigns a low probability to every class. Moreover, the model-centric manifold also contains sharp glyphs that do not represent any digit.

The main contributions of the paper are:
1. the definition of the local data matrix $G(x, w)$ at a point x of the data domain and for a given model w, and the study of its properties;
2. the proof that the subspace spanned by the eigenvectors with non-zero eigenvalue of the local data matrix $G(x, w)$ can be interpreted as the tangent space of a Riemannian manifold;
3. the visualization of the model-centric data manifold through paths, obtained via experiments on MNIST.

Organization of the paper. In Sec. 2, we review the fundamentals of information geometry using a novel perspective that aims at facilitating the comprehension of the key concepts of the paper. We introduce the *local data matrix* $G(x, w)$ and we summarize its properties in Prop. 2.1. In Sec. 3, we show that, through the local data matrix, under some mild hypotheses, the data domain foliates as a disjoint union of leaves, which are all Riemannian submanifolds of $\mathbb{R}^n$, with metric given via $G(x, w)$. In Sec. 4, we provide evidence that all our dataset lies on one leaf of the foliation and moving along directions orthogonal to the data leaf amounts to adding noise to our data.

Related works. In Achille et al. (2018) the authors discuss the information content of the Fisher matrix, and they show that such content changes during the training of a neural network, decreasing rapidly towards the end of it. This shows that the Fisher-Rao metric acquires importance during the training phase only (see also Kirkpatrick et al. (2017)). More on the geometry of the parameter space, as a Riemannian manifold, is found in Sommer & Bronstein (2020). In this paper, we take an analog of the Fisher-Rao metric, but on the data domain. In our examples, we see a phenomenon similar to the one observed in Achille et al. (2018): the trace of the local data matrix decreases rapidly, as the model completes its training. Other metrics on datasets were suggested, for example see Dukler et al. (2019) and refs. within, but with different purposes. Here, our philosophy is the same as in Bergomi et al. (2019): we believe that data itself is not equipped with a geometric structure, but such structure emerges only when the model views data, with a given classification task.

2 Information Geometry

Here we collect some results pertaining to information geometry (Amari, 1998; Nielsen, 2018), using a novel, convenient perspective. Let $p(y|x, w)$ be a discrete probability distribution on C classification labels, i.e. $p(y|x, w) = (p_i(y|x, w))_{i=1,\dots,C}$, $x \in \Sigma \subset \mathbb{R}^n$, $w \in \mathbb{R}^d$. In the applications, the x's represent the data belonging to a certain dataset Σ, while w are the learning parameters, i.e.

the parameters of the empirical model. As we are going to see in our discussion later on, it is fruitful to treat the two sets of variables x and w on equal grounds.

In order to give some context to our treatment, we define, following Amari (1998) Sec. 3, the *information loss* $I(x,w) = -\log(p(y|x,w))$ and the *loss function* $L(x,w) = \mathbb{E}_{y\sim q}[I(x,w)]$. Typically $L(x,w)$ is used for practical optimizations, where we need to compare the model output distribution $p(y|x,w)$ with a certain known true distribution $q(y|x)$. We may also view $L(x,w)$ as the Kullback-Leibler divergence up to the constant $-\sum_i q_i(y|x)\log q_i(y|x)$, irrelevant for any optimization problem:

$$\begin{aligned} L(x,w) = \mathbb{E}_{y\sim q}[-\log(p(y|x,w))] &= \sum_{i=1}^{C} q_i(y|x)\log\frac{q_i(y|x)}{p_i(y|x,w)} - \sum_{i=1}^{C} q_i(y|x)\log q_i(y|x) = \\ &= D_{\mathrm{KL}}(q(y|x)||p(y|x,w)) - \sum_{i=1}^{C} q_i(y|x)\log q_i(y|x) \end{aligned} \tag{1}$$

A popular choice for $p(y|x,w)$ in deep learning classification algorithms is

$$p_i(y|x,w) = \mathrm{softmax}(s_1(x,w),\dots,s_C(x,w))_i = \frac{e^{s_i(x,w)}}{\sum_{j=1}^{C} e^{s_j(x,w)}}, \tag{2}$$

where $s(x,w) \in \mathbb{R}^C$ is a score function determined by parameters w. From such $p(y|x,w)$ we derive the cross-entropy with softmax loss function:

$$L(x,w) = \mathbb{E}_{y\sim q}[I(x,w)] = \mathbb{E}_{y\sim q}[-\log p(y|x,w)] = -s_{y_x}(x,w) + \log\sum_{j=1}^{C} e^{s_j(x,w)}, \tag{3}$$

where $L(x,w)$ is computed with respect to the probability mass distribution $q(y|x)$ assigning 1 to the correct label y_x of our datum x and zero otherwise.

Going back to the general setting, notice that: $\mathbb{E}_{y\sim p}[I(x,w)] = 0$. In fact,

$$\mathbb{E}_{y\sim p}[\nabla_w(\log(p(y|x,w))] = \sum_{i=1}^{C} p_i(y|x,w)\nabla_w \log p_i(y|x,w) = \sum_{i=1}^{C} \nabla_w p_i(y|x,w) = \nabla_w 1 = 0. \tag{4}$$

Let us now define the following two matrices:

$$F(x,w) = \mathbb{E}_{y\sim p}[\nabla_w(\log(p(y|x,w)) \cdot \nabla_w(\log(p(y|x,w))^T] \tag{5}$$

$$G(x,w) = \mathbb{E}_{y\sim p}[\nabla_x(\log(p(y|x,w)) \cdot \nabla_x(\log(p(y|x,w))^T] \tag{6}$$

We call $F(x,w)$ the *local Fisher matrix* at the datum x and $G(x,w)$ the *local data matrix* given the model w. The Fisher matrix (Amari, 1998) is obtained as $F(w) = \mathbb{E}_{x\sim\Sigma}[F(x,w)]$ and it gives information on the metric structure of the space of parameters. Similarly, we can reverse our perspective and see how $G(x,w)$, allows us to recognize some structure in our dataset.

The following observations apply to both $F(x,w)$ and $G(x,w)$ and provide the theoretical cornerstone of Sec. 3.

Proposition 2.1. *Let the notation be as above. Then:*

1. $\ker F(x,w) = \mathrm{span}\{\nabla_w \log p_i(y|x,w) | 1 \le i \le C\}^{\perp}$;
   $\ker G(x,w) = \mathrm{span}\{\nabla_x \log p_i(y|x,w) | 1 \le i \le C\}^{\perp}$.
2. $\mathrm{rank}\, F(x,w) < C, \quad \mathrm{rank}\, G(x,w) < C.$

Proof. We prove the results for $F(x,w)$, the proofs for $G(x,w)$ are akin.
1. We show that $\ker F(x,w) \subseteq \mathrm{span}\{\nabla_w \log p_i(y|x,w) | 1 \le i \le C\}^{\perp}$:

$$\begin{aligned} u \in \ker F(x,w) \Rightarrow u^T F(x,w) u = 0 \Rightarrow \mathbb{E}_{y\sim p}\left[\langle \nabla_w \log p(y|x), u\rangle^2\right] = 0 \\ \Rightarrow \langle \nabla_w \log p_i(y|x), u\rangle = 0 \quad \forall\, i = 1,\dots,C. \end{aligned} \tag{7}$$

On the other hand, if $u \in \mathrm{span}\{\nabla_w \log p_i(y|x,w) | 1 \le i \le C\}^{\perp}$ then $u \in \ker F(x,w)$:

$$F(x,w)u = \mathbb{E}_{y\sim p}\left[\nabla_w \log p_i(y|x)\langle \nabla_w \log p_i(y|x), u\rangle\right] = 0. \tag{8}$$

2. From 2.1.1, we know that $\mathrm{rank}\, F(x,w) \le C$. Equation (4) allows us to conclude the proof. □

3 THE MODEL VIEW ON THE DATA MANIFOLD

We now turn to examine some properties of the matrices $F(x, w)$ and $G(x, w)$ defined in the previous section, that will enable us to discover a manifold structure on the portion of $\mathbb{R}^n$ occupied by our data set.

We recall that, given a perturbation of the weights w, the Kullback-Leibler divergence, gives, to second-order approximation, the following formula:

$$D_{\mathrm{KL}}(p(y|x, w + \delta w)||p(y|x, w)) = (\delta w)^T F(x, w)(\delta w) + \mathcal{O}(||\delta w||^3) \tag{9}$$

The Fisher matrix $F(w) = \mathbb{E}_{x\sim\Sigma}[F(x, w)]$, when non degenerate, is then an effective metric on the parameter space. It allows us to measure, according to a certain step δw, when we reach a stable predicted probability and thus the end of model training (see Martens (2020) and refs. within). It must be however noted that $F(w)$ retains its information content only *away* from the trained model, that is, well before the end of the training phase (see Achille et al. (2018) for an empirical validation of such statements). We are going to see, with our experiments in Sec. 4, that a similar phenomenon occurs for $G(x, w)$.

We now turn to the local data matrix $G(x, w)$, thus interpreting equation (9) in the data domain $\mathbb{R}^n$. For a perturbation δx of the data x, we have, up to second order approximation:

$$D_{\mathrm{KL}}(p(y|x + \delta x, w)||p(y|x, w)) = (\delta x)^T G(x, w)(\delta x) + \mathcal{O}(||\delta x||^3) \tag{10}$$

This equation suggests to view $G(x, w)$ as a metric on the data domain. However, because of its low rank (see Prop. 2.1), we need to restrict our attention to the subspace $\ker(G(x, w))^\perp$, where $G(x, w)$ is non-degenerate. $G(x, w)$ allows us to define a *distribution* $\mathcal{D}$ on $\mathbb{R}^n$. In general, in differential geometry, we call distribution on $\mathbb{R}^n$ an assignment:

$$x \mapsto \mathcal{D}_x \subset \mathbb{R}^n, \qquad \forall x \in \mathbb{R}^n$$

where $\mathcal{D}_x$ is a vector subspace of $\mathbb{R}^n$ of a fixed dimension k (see Appendix A for more details on this notion in the general context).

Assume now $G(x, w)$ has constant rank; as we shall see in our experiments, this is the case for a non fully trained model (see Sec. 4 for more details). We thus obtain a *distribution* $\mathcal{D}$:

$$x \mapsto \mathcal{D}_x = \ker(G(x, w))^\perp \subset \mathbb{R}^n \tag{11}$$

Equation (10) tells us that, if we move along the directions of $\ker(G(x, w))$, the probability distribution $p(y|x, w)$ is constant (up to a second order approximation) while our data is changing. Those are the vast majority of the directions, since $\operatorname{rank} G(x, w) < C$ and typically $C << n$, hence, we interpret them as the *noise directions*. On the other hand, if we move from a data point x along the directions in $\ker(G(x, w))^\perp$, data will change along with the probability distribution $p(y|x, w)$ associated with it. These are the directions in which the model moves with confidence; we are going to clarify this key point in Sec. 4.

We now would like to see if our distribution (11) defines a submanifold structure in a neighbourhood of a data point x, so that the tangent space at x is $\mathcal{D}_x = \ker(G(x, w))^\perp$. In this way, moving along the directions in $\mathcal{D}$ at each point, will produce a path laying in the submanifold through x.

The answer to this question comes easily through Frobenius theorem, which we state in the Appendix A and we recall here in the version we need.

Frobenius Theorem. *Let $x \in \mathbb{R}^n$ and let $\mathcal{D}$ be a distribution in $\mathbb{R}^n$. Assume that in a neighbourhood U of x:*

$$[X, Y] \in \mathcal{D}, \qquad \forall\, X, Y \in \mathcal{D}. \tag{12}$$

Then, there exists a (local) submanifold $N \subset \mathbb{R}^n$, $x \in N$, such that $T_zN = \mathcal{D}_z$, for all $z \in N$.

It is not reasonable to expect that a general classifier satisfies the involutive property (12), however it is remarkable that for a large class of classifiers, namely deep ReLU neural networks, this is the case, with p given by softmax as in equation (2).

Theorem 3.1. *Let w be the weights of a deep ReLU neural network classifier, p given by softmax, $G(x,w)$ the local data matrix. Assume $G(x,w)$ has constant rank. Then, there exists a local submanifold $N \subset \mathbb{R}^n$, $x \in N$, such that its tangent space at z, $T_zN = \ker(G(z,w))^\perp$ for all $z \in N$.*

Proof. By Frobenius Theorem, we need to check the involutivity property (12) for the distribution $x \mapsto \mathcal{D}_x = \ker(G(x,w))^\perp$ on $\mathbb{R}^n$. Since by Prop. 2.1,

$$\mathcal{D}_x = \ker(G(x,w))^\perp = \text{span}\{\nabla_x \log p_k(y|x,w) \,|\, 1 \leq k \leq C\}$$

we only need to show that:

$$[\nabla_x \log p_i(y|x,w), \nabla_x \log p_j(y|x,w)] \in \text{span}\{\nabla_x \log p_k(y|x,w) \,|\, 1 \leq k \leq C\}$$

By standard computations, we see that:

$$\begin{aligned}[\nabla_x \log p_i(y|x,w), \nabla_x \log p_j(y|x,w)] \quad &= \mathbb{H}(\log p_i(y|x,w))\nabla_x \log p_j(y|x,w) + \\ &-\mathbb{H}(\log p_j(y|x,w))\nabla_x \log p_i(y|x,w),\end{aligned} \tag{13}$$

where $\mathbb{H}(f)$ denotes the Hessian of a function f. With some easy calculations, we have:

$$\mathbb{H}(\log p_i(y|x,w)) = \frac{\mathbb{H}(p_i(y|x,w))}{p_i(y|x,w)} - \nabla_x \log p_i(y|x,w) \cdot (\nabla_x \log p_i(y|x,w))^T \tag{14}$$

and

$$\nabla_x p_i(y|x,w) = \sum_{k=1}^{C} p_i(y|x,w)(\delta_{ik} - p_k(y|x,w))\nabla_x s_k, \tag{15}$$

where s is the score function and we make use of the fact $p_i(y|x,w)$ is given by softmax. Notice $\partial_{s_k} p_i(y|x,w) = p_i(y|x,w)(\delta_{ik} - p_k(y|x,w))$, where δ_{ik} is the Kronecker delta. By (14) and (15):

$$\mathbb{H}(p_i(y|x,w)) = \text{Jac}(\nabla_x p_i(y|x,w)) = \sum_{k=1}^{C} \nabla_x \left[p_i(y|x,w)(\delta_{ik} - p_k(y|x,w))\right] (\nabla_x s_k)^T, \tag{16}$$

where we use the fact that s_k is piecewise linear so that its Hessian is zero, where it is defined. Hence:

$$\mathbb{H}(p_i(y|x,w)) = \frac{\nabla_x p_i(y|x,w)(\nabla_x p_i(y|x,w))^T}{p_i(y|x,w)} - p_i \sum_{k=1}^{C} \nabla_x p_k(y|x,w)(\nabla_x s_k)^T. \tag{17}$$

Now, in view of (13), using (14) and (17) we compute the expression:

$$\begin{aligned}\mathbb{H}(\log p_i(y|x,w)) = \frac{\nabla_x p_i(y|x,w)}{p_i(y|x,w)} \left(\frac{\nabla_x p_i(y|x,w)}{p_i(y|x,w)}\right)^T - \sum_{k=1}^{C} \nabla_x p_k(y|x,w)(\nabla_x s_k)^T + \\ - \nabla_x \log p_i(y|x,w) \cdot (\nabla_x \log p_i(y|x,w))^T = -\sum_{k=1}^{C} \nabla_x p_k(y|x,w)(\nabla_x s_k)^T.\end{aligned} \tag{18}$$

Hence:

$$\mathbb{H}(\log p_i(y|x,w))\nabla_x \log p_j(y|x,w) = -\sum_{k=1}^{C} \nabla_x p_k(y|x,w)(\nabla_x s_k)^T \nabla_x \log p_j(y|x,w).$$

Since $(\nabla_x s_k)^T \nabla_x \log p_j(y|x,w)$ is a scalar, we obtain $\mathbb{H}(\log p_i(y|x,w))\nabla_x \log p_j(y|x,w)$ is a linear combination of vectors $\nabla_x p_k(y|x,w)$, and therefore a linear combination of $\nabla_x \log p_k(y|x,w)$. The same holds for $[\nabla_x \log p_i(y|x,w), \nabla_x \log p_j(y|x,w)]$ as we wanted to show. □

Through the application of Frobenius theorem, Theorem 3.1 gives us a *foliation* of the data domain $\mathbb{R}^n$: $\mathbb{R}^n$ decomposes into the disjoint union of $C-1$ dimensional submanifolds, whose tangent space at a fixed $x \in \mathbb{R}^n$ is $\mathcal{D}_x = \ker(G(x,w))^\perp$. If $x \in \Sigma$, our dataset, x determines a unique submanifold corresponding to the leaf of the foliation through x. As we shall see in Sec. 4, we can

move from a point x in our dataset Σ to another point x' also in Σ with an *horizontal* path, that is a path tangent to $\mathcal{D}_x$, hence laying on the leaf of x and x'. So, in our experiments, we shall see that all the dataset belongs to a single leaf, which we call the *data leaf* $\mathcal{L}$. Our model, through the local data matrix $G(x, w)$, enables us to move on the low-dimensional submanifold $\mathcal{L}$, to which all of our dataset belongs. Of course not all the points of $\mathcal{L}$ correspond to elements of the dataset; however as we show in the experiments, on most points of $\mathcal{L}$ the model gives predictions compatible with human observers.

We also notice that each leaf of our foliation comes naturally equipped with a metric given at each point by the matrix $G(x, w)$, restricted to the subspace $\ker(G(x, w))^{\perp}$, which coincides with the tangent space to the leaf, where $G(x, w)$ is non-degenerate. We end this section with an observation, comparing our approach to the geometry of the data domain, with the parameter space.

Remark. Equation (9) provides a metric to the parameter space $\mathbb{R}^d$, motivating our approach to the data domain. For each $w \in \mathbb{R}^d$, we can define, as we did for the data domain, a distribution $w \mapsto \mathcal{D}'_w := \ker(F(w))^{\perp}$, using the Fisher matrix. However, it is easy to see empirically that this distribution is *not involutive*, i.e. there is no foliation and no submanifold corresponding to it.

4 EXPERIMENTS

The definition of distribution in Sec. 3 and the consequent results require the rank of $G(x, w)$ to be constant on every point x for a certain parameter configuration w. Thus, we need to verify experimentally whether this hypothesis holds in practice. All our experiments are performed using a simple CNN classifier trained on the MNIST dataset (LeCun et al., 1998).

We observe that during the training of the CNN, the rank of $G(x, w)$ is stable and equal to $C - 1$. That changes at the end of the training, when the model reaches convergence. At that point, the output class probability distribution on most images is highly unbalanced and sparse and all the eigenvalues of $G(x, w)$ tend to 0. Figure 1 shows the decreasing trend of the mean trace of $G(x, w)$ on a batch during the training. The plot is similar to the ones reported in Achille et al. (2018) for the Fisher information matrix and it tells us that the local data matrix $G(x, w)$ loses its informative content at the end of training as well. This observation motivates us to perform subsequent experiments on a partially trained model and with data points where every output class probability is lower than 0.99.

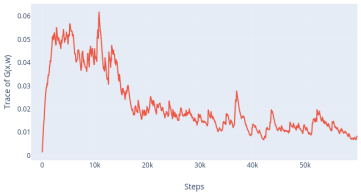

Figure 1: Mean trace of $G(x, w)$ during training

Since a complex model reaches convergence on MNIST digit classification too fast, we have resorted to a very small CNN. The network used in the following experiments has 18,982 parameters and ReLU activation function. It is similar to LeNet, but with 4 channels in both the two convolutional layers and 32 hidden units in the fully-connected layer. We train this network with SGD for just one epoch to prevent model convergence. Such a neural network classifier satisfies all the hypothesis of Theorem 3.1, so it views the data domain $\mathbb{R}^n$ as a *foliation*. However, this result does not give us any clue about the characterization of leaves.

To understand the distinguishing factors of a leaf, we move a data point x across leaves and we inspect its evolution. We start from an image in MNIST test set and we let it evolve moving orthogonally to leaves. In order to get further and further away, we use a fixed random direction and at every step we project it on the space orthogonal to the current leaf. From the previous section, we know that such a space coincides with $\ker G(x, w)$.

We observe in Figure 2, that the noise in the images increases steadily. Moreover, we can see that, as predicted by equation (10), model predictions remain very certain even when the digit is indistinguishable for the human eye.

The previous experiment makes us speculate that a leaf is characterized by a constant amount of noise. If that is the case, all valid data should reside on the same leaf characterized by the absence of noise, the data leaf. To verify our intuition we follow *horizontal* paths connecting two images from MNIST test set. If it is possible to join two inputs with a horizontal path, then we know that

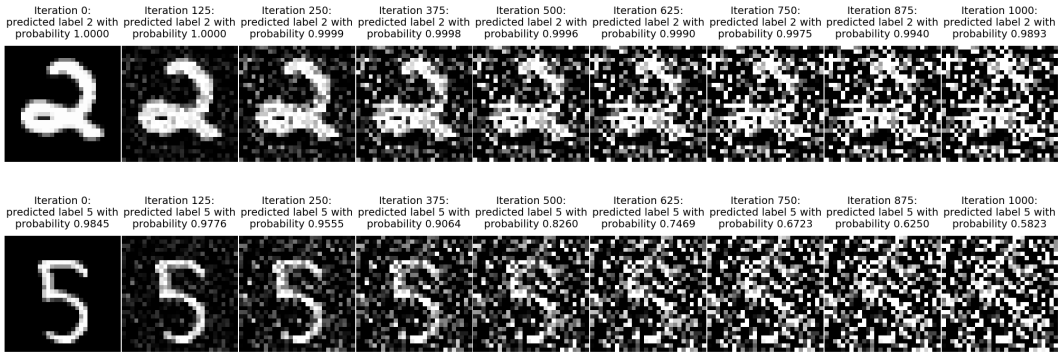

Figure 2: Paths across leaves of the foliation from a valid image.

those points are on the same leaf. In our case a horizontal path is tangent to the distribution $\mathcal{D}$ described in the previous section, i.e. $\mathcal{D}_x = (\ker G(x,w))^{\perp} = \text{span}\{\nabla_x \log p_i(y|x,w) | 1 \leq i \leq C\}$ from Proposition 1. We remark that $\mathcal{D}_x$ coincides with the row space of the Jacobian matrix $\text{Jac}_x \log p(x,w)$.

The algorithm to approximate a path from a source point to a destination point is a simplification of Riemannian gradient descent (Gabay, 1982) using the euclidean distance from the destination point as loss function. Since we do not have an explicit characterization of the leaf, we cannot perform the retraction step commonly used in optimization algorithms on manifold. To circumvent this problem, we normalize the gradient vector and use a small step size of 0.1 in our experiments. The iterative algorithm is reported in Algorithm 1.

Algorithm 1: Find a horizontal path between source and destination points.

Input: s: source; d: destination; α: step size, T: number of iterations; w: model parameters.

$x_0 \leftarrow s$;
for $t = 1, \ldots, T$ **do**
  $j \leftarrow \text{Jac}_x \log p(x_{t-1}, w)$ (Calculate Jacobian of $\log p(x_{t-1}, w)$ w.r.t. x_{t-1});
  $v \leftarrow \text{projection}(d - x_{t-1}, j)$ (Project the gradient of $||d - x_{t-1}||^2$ on $\mathcal{D}_{x_{t-1}}$);
  $x_t \leftarrow x_{t-1} + \alpha \frac{v}{||v||}$ (Update x_{t-1} to obtain the next point x_t on the horizontal path);
return $\{x_t\}_{t=0,\ldots,T}$

Figure 3 shows the results of the application of Algorithm 1 on some pairs of images in MNIST test set. We observe that it is possible to link very different images with an horizontal path, confirming our conjecture about the existence of the data leaf. This experiment definitely shows that the model sees the data on the same manifold.

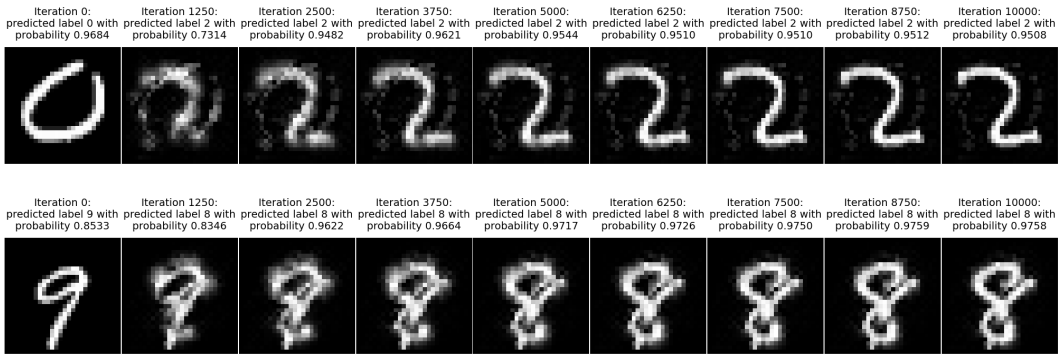

Figure 3: Horizontal paths between two images in MNIST test set.

The observation of the paths on the data leaf gives us a novel point of view on the generalization property of the model. First of all, while we could expect to find all training data on the same leaf, it is remarkable that test data are placed on the same leaf too. Furthermore, we observe that the data leaf is not limited to digits and transition points between them. In fact, we can find paths connecting valid images with images of non-existent digits similar to letters, as shown in Figure 4. That fact suggests that the model-centric data manifold is somewhat more general than the classification task on which the model is trained.

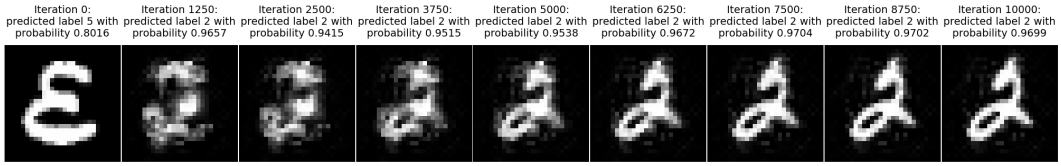

Figure 4: Horizontal path between a mirrored image and a valid image from MNIST test set.

Our final experiment gives us another remarkable confirmation of our theoretical findings: it is impossible to link with a horizontal path a noisy image outside the data leaf and a valid data image. Figure 5 shows that even after thousands of iterations of Algorithm 1, it is impossible to converge to the destination point. Indeed, the noise is preserved along the path and the noise pattern stabilizes after 5000 iterations. The final point reached is a noisy version of the actual destination point.

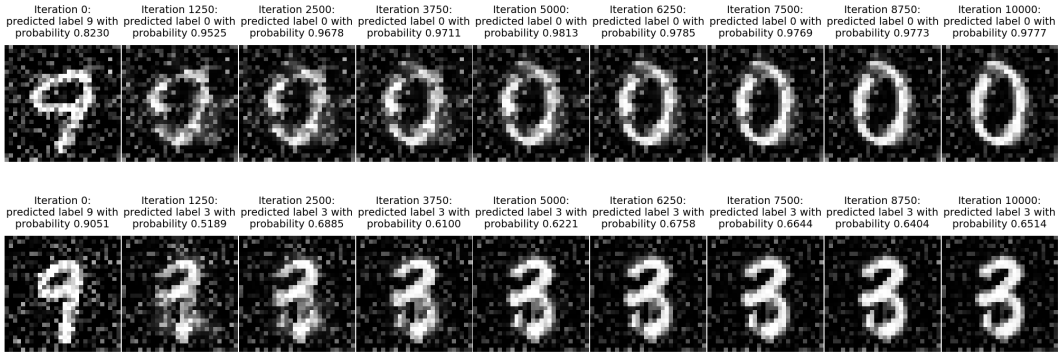

Figure 5: Horizontal paths unable to reach a valid image in MNIST test set from a noisy image.

All those experiments confirm that there exists a model-centric data manifold, the data leaf of the foliation. In addition they exhibit that the leaves in the foliation of the data domain are characterized by the noise content.

5 Conclusions

In this paper, we introduce the local data matrix, a novel mathematical object that sheds light on the internal working of a neural network classifier. We prove that the model organizes the data domain according to the geometric structure of a foliation. Experiments show that valid data are placed on the same leaf of the foliation, thus the model sees the data on a low-dimensional submanifold of the data domain. Such submanifold appears more general than the model itself because it includes meaningless but visually similar images together with training and test data.

In the future, we aim to characterize the data leaf and to study the Riemannian metric given by the local data matrix. If we could analytically characterize the leaves of the foliation by the degree of noise, we could distinguish noisy data from valid examples. That can be used in the inference phase to exclude examples on which model predictions are not reliable. Furthermore, it could pave the way to the creation of novel generic denoising algorithms applicable to every kind of data.

REFERENCES

Ralph Abraham, Jerrold E Marsden, and Tudor Ratiu. *Manifolds, tensor analysis, and applications*, volume 75. Springer Science & Business Media, 2012.

Alessandro Achille, Matteo Rovere, and Stefano Soatto. Critical learning periods in deep networks. In *International Conference on Learning Representations*, 2018.

Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.

Martín Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. In *International Conference on Learning Representations*, 2017.

Mattia G Bergomi, Patrizio Frosini, Daniela Giorgi, and Nicola Quercioli. Towards a topological–geometrical theory of group equivariant non-expansive operators for data analysis and machine learning. *Nature Machine Intelligence*, 1(9):423–433, 2019.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12:2493–2537, 2011.

Yonatan Dukler, Wuchen Li, Alex Lin, and Guido Montufar. Wasserstein of wasserstein loss for learning generative models. volume 97 of *Proceedings of Machine Learning Research*, pp. 1716–1725. PMLR, 2019.

Daniel Gabay. Minimizing a differentiable function over a differential manifold. *Journal of Optimization Theory and Applications*, 37(2):177–219, 1982.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014.

Durk P. Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in neural information processing systems*, pp. 10215–10224, 2018.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

James Martens. New insights and perspectives on the natural gradient method. *Journal of Machine Learning Research*, 21(146):1–76, 2020.

Frank Nielsen. An elementary introduction to information geometry. *arXiv preprint arXiv:1808.08271*, 2018.

Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2017. doi: 10.23915/distill.00007. https://distill.pub/2017/feature-visualization.

William Peebles, John Peebles, Jun-Yan Zhu, Alexei A. Efros, and Antonio Torralba. The hessian penalty: A weak prior for unsupervised disentanglement. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2020.

Stefan Sommer and Alex M Bronstein. Horizontal flows and manifold stochastics in geometric deep learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

Loring W. Tu. *An Introduction to Manifolds*. Universitext, Springer, 2008.

Aaron Van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. In *Advances in neural information processing systems*, pp. 4790–4798, 2016.

A APPENDIX

For the reader's convenience, we collect here few facts regarding differentiable manifolds, for more details see Abraham et al. (2012), Tu (2008).

Let M be a differentiable manifold. Its tangent space T_xM at a point $x \in M$ can be effectively defined as the vector space of derivations at x, so $T_xM = \{\sum_i a^i \partial_i\}$, after we choose a chart around x. A *vector field* X on M is a function $x \mapsto X_x \in T_xM$ assigning to any point of $x \in M$ a tangent vector X_x. If $TM = \coprod_{x \in M} T_xM$ (disjoint union) denotes the *tangent bundle*, we also call the vector field X a *section* of TM.

Since T_xM is the vector space of derivations on differentiable functions defined on a neighbourhood of x, we can write $X_x(f)$; this is the result of the application of the derivation $X_x \in T_xM$ to the function $f \in C^\infty_M(U)$, where $C^\infty_M(U)$ denotes the differentiable functions $f : U \longrightarrow \mathbb{R}$, $x \in U$ open in M. As x varies in U $X_x(f)$ defines a function, which we denote with $X(f)$.

We call $\chi(M)$ the *smooth* vector fields on M, that is, those vector fields expressed in local coordinates as $X_x = \sum_i a_i(x)\partial_i$, with $a_i : U \longrightarrow \mathbb{R}$ smooth functions, $x \in U$, U open in M.

We can define a *bracket* on the space of vector fields as follows:

$$[X, Y](f) := X(Y(f)) - Y(X(f)), \qquad f \in C^\infty_M(M), \quad X, Y \in \chi(M)$$

$[\,,\,]$ is bilinear and satisfies:
1. $[X, Y] = -[Y, X]$ (antisymmetry);
2. $[X, [Y, Z]] + [Y, [Z, X]] + [Z, [X, Y]] = 0$ (Jacoby identity).

We are ready to define *distributions*, which play a key role into our treatment.
Let M be a differentiable manifold. We define a *distribution* $\mathcal{D}$ of rank k an assignment:

$$x \longrightarrow \mathcal{D}_x \subset T_xM, \qquad \forall x \in M$$

where $\mathcal{D}_x$ is a subspace of T_xM of dimension k. We say that $\mathcal{D}$ is *smooth* if at each point $x \in M$, there exists an open set U, $x \in U$, $\mathcal{D}_y$ is spanned by smooth vector fields on U for all $y \in U$.

We say that a distribution is *integrable* if for any $x \in M$ there exists a local submanifold $N \subset U$, U open in M, $x \in N$, called a *local integrable manifold* such that $T_zN = \mathcal{D}_z$ for all $z \in N$. When such N exist globally, we say we have a *foliation* of M, that is we can write M as the disjoint union of integrable submanifolds. For example in $\mathbb{R}^3$, the distribution:

$$P = (x, y, z) \mapsto \mathcal{D}_P = (\text{span}\{(x, y, z)\})^\perp$$

is integrable and foliates the space $\mathbb{R}^3$ as the disjoint union of spheres.

We say that a distribution is *involutive* if for all vector fields $X, Y \in \mathcal{D}$ we have $[X, Y] \in \mathcal{D}$.

Frobenius Theorem establishes an equivalence between these two key properties of a distribution, namely integrability and involutivity, thus giving us an effective method to establish when a distribution gives to a foliation of M into disjoint submanifolds.

Theorem A.1. Frobenius Theorem *Let M be a differentiable manifold and $\mathcal{D}$ a distribution on M. Then $\mathcal{D}$ is involutive if and only if it is integrable. If this occurs, then there exists a foliation on M, given by the integrable submanifolds of $\mathcal{D}$.*

Proof. See Abraham et al. (2012) Sec. 4.4. □