Connecting Ideas About Neural Network Generalization

Thomas Walker

July 2023

Contents

Abstract

There are three main components of neural networks that are used to understand their generalization capacities. These components are the training data, the training algorithm and the architecture. Some approaches deal exclusively with one of these components while other approaches aim to capture multiple components. For example, information-theoretic approaches focus on the training data and how information is transferred to the representation of the network. On the other hand, Bayesian machine learning is a framework for investigating the learning algorithm. Forming data-inspired priors was an extension of this framework to incorporate training data to get tighter bounds on the generalization error. To get a comprehensive picture of neural network generalization we ought to incorporate each of these three components.

1 Stiffness

Recall that stiffness from [1] investigates the quantity

$$\bar{g} = \nabla_{\mathbf{w}} l(h_{\mathbf{w}}(x), y),$$

for a functional approximation $h_{\mathbf{w}}$, parameterized by a trainable parameter $\mathbf{w}$. Where $l(h_{\mathbf{w}}(x), y))$ is the loss function.

Definition 1.1. *For two data points* (x_1, y_1) *and* (x_2, y_2) *define the sign stiffness to be*

$$S_{\text{sign}}\left((x_1, y_1), (x_2, y_2); f\right) = \mathbb{E}\left(\text{sign}\left(\bar{g}_1 \cdot \bar{g}_2\right)\right).$$

Definition 1.2. *For two data points* (x_1, y_1) *and* (x_2, y_2) *define the cosine stiffness to be*

$$S_{\cos}\left((x_1, y_1), (x_2, y_2); f\right) = \mathbb{E}\left(\cos\left(\bar{g}_1 \cdot \bar{g}_2\right)\right),$$

where

$$\cos\left(\bar{g}_1 \cdot \bar{g}_2\right) = \frac{\bar{g}_1 \cdot \bar{g}_2}{|\bar{g}_1||\bar{g}_2|}.$$

Stiffness captures information from the training data as the gradient of the loss function is evaluated at data points. It also captures information about the learning algorithm as the gradient of the loss function with respect to the parameters determines the update step of stochastic gradient descent. In the subsequent section, I look to investigate how stiffness can be related to the architecture of the neural network.

1.1 Distance

For a data point, we can investigate the stiffness in relation to the examples within a neighbourhood defined by some metric. For a clustering task, we can consider Euclidean distance to identify the boundaries of a cluster. Refer to Figure 1 for a visualization of this for a two-dimensional example.

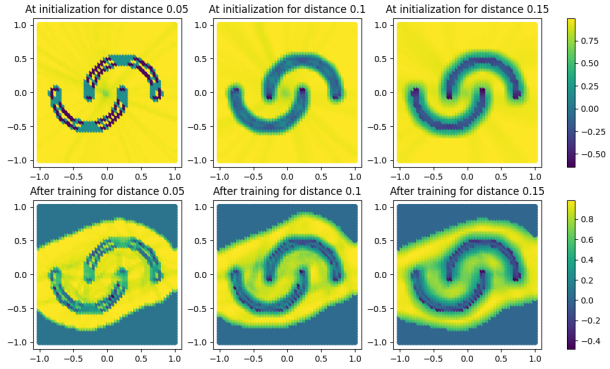

Figure 1: The stiffness of points within a neighbourhood can be used to identify the learned boundary of a classification problem.

1.2 Layers

Instead of representing the parameters as a singular vector, we can decompose them to reflect the architecture of the network. In the following we conduct similar analyses but with the gradient taken with respect to a subset of the parameters to identify how stiffness evolves through the layers of a network. In the following, we consider the stiffness of data points in relation to a balanced sample of data points from the categories of the dataset. In Figure 2 we compare how the stiffness varies through the layers of a trained and untrained network. Then, in Figure 3 we look at how stiffness varies through the layers for networks of different architectures.

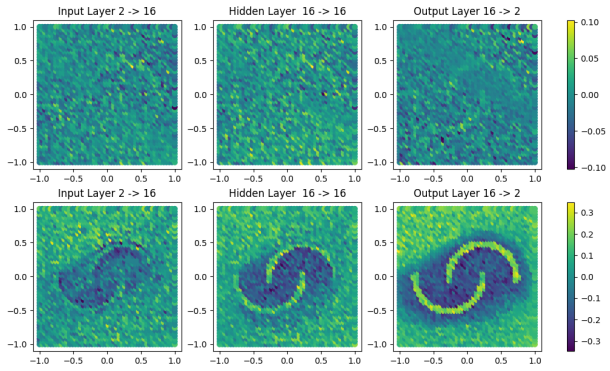

Figure 2: Average stiffness between an example and a random sample from the training data containing samples from both categories.

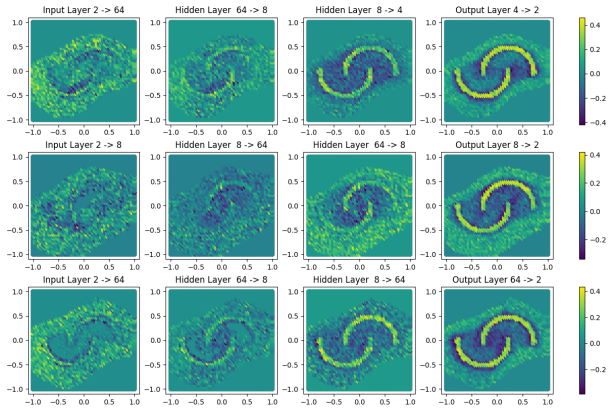

Figure 3: This shows how the stiffness of examples changes through the layers of networks with different numbers of hidden units.

1.3 Activation Functions

In the original experiment, ReLU activation functions connected the layers of the networks. We can instead use $\tanh$ and LeakyReLU activation and observe how this affects the stiffness through the layers. My intention here is to understand whether the conclusions of [2] are present when neural networks are investigated from this perspective. Figure 4 shows the results for $\tanh$ activation, whereas, Figure 5 shows the results for networks with LeakyReLU activation.

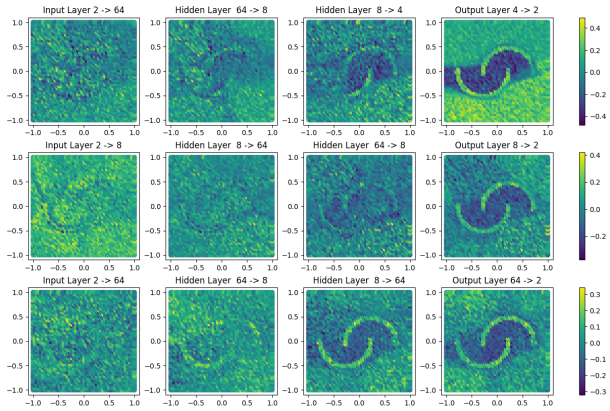

Figure 4: Stiffness through network layers connected by $\tanh$ activations

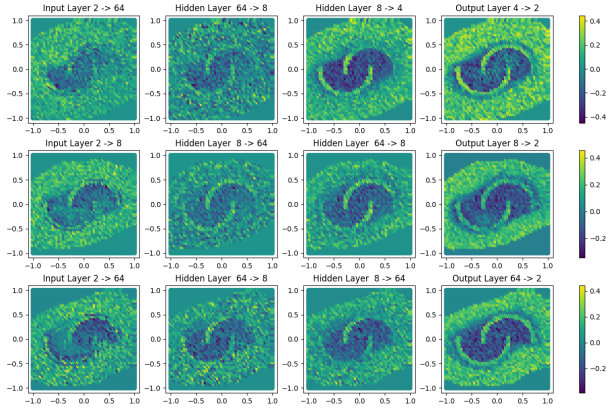

Figure 5: Stiffness through network layers connected by LeakyReLU activations

1.4 Depth

By keeping the width of our networks constant we can investigate how stiffness varies through the layers as the depth of the network is changed. The result of these investigations can be seen in Figure 6.

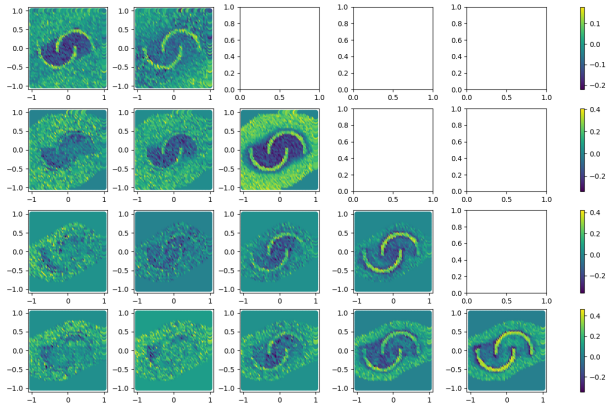

Figure 6: Stiffness between layers of a ReLU network with constant width of 16 hidden units.

1.5 Conclusions

We now compare our results with the conclusions made by [2]. It is clear that both ReLU and LeakyReLU activations extenuate the features of the dataset faster and with greater precision than $\tanh$ activations. Which supports the conclusions of [2]. ReLU was seen to cause the largest topological changes, and in our experimentation, we also see that ReLU identifies the features of the data set more clearly than the other activation functions. The effect of the bottleneck identified in [2] was similarly seen in our results. In [2] by changing the depth of a neural network of constant width it was shown that the burden of inducing topological changes was shifted toward the later layers. This is equally seen in my work, as the features start to become prominent in the latter layers of networks of increased depth.

2 Linking Bayesian Machine Learning and Information Theory

The Bayesian and Information-theoretic perspectives bound the generalization of learning algorithms on a notion of similarity between distributions. For the Bayesian framework, the similarity is quantified using KL divergence of the prior and posterior distributions. Whereas, the information-theoretic framework focuses on the mutual information between the training data distribution and the distribution of the learned classifier. In the survey there was a hint of how these two frameworks could be reconciled, however, here we make the connection explicit.

2.1 Notation

In the following, we will consider a hypothesis class $\mathcal{H}$ of neural networks parameterised by weights $\mathbf{w} \in \mathcal{W}$. The data will be assumed to have come from a distribution $\mathcal{D}$ defined over the sample space $\mathcal{X} \times \mathcal{Y} = \mathcal{Z}$, which we can write as $\mathcal{D} \in \mathcal{M}(\mathcal{Z})$. The loss function used to train the network will be denoted l and the learned parameter value will be $\hat{\mathbf{w}} \in \mathcal{W}$ which defines the classifier $h_{\hat{w}}$. To train the network we will assume we have an $\mathrm{i.i.d}$ sample S from $\mathcal{D}^m$, which will be denoted $S \sim \mathcal{D}^m$. The loss function quantifies the quality of a particular hypothesis on the sample space, formally this means that $l : \mathcal{Y} \times \mathcal{Y} \to [0, 1]$. We will let

$$R(\mathbf{w}) = \mathbb{E}_{z\sim\mathcal{D}}(l(h_{\mathbf{w}}(x), y)),$$

and

$$\hat{R}(\mathbf{w}) = \frac{1}{m}\sum_{i=1}^{m} l(h_{\mathbf{w}}(x_i), y_i).$$

2.2 PAC-Bayes Results

To apply the Bayesian machine learning framework we suppose we have a prior distribution π taken from a set of distributions $\mathcal{M}(\mathcal{W})$ on the space of parameters which in turn induces the set of distributions defined on the hypothesis space, denoted $\mathcal{M}(\mathcal{H})$. This prior belief is then updated via the learning algorithm to a posterior distribution $\rho \in \mathcal{M}(\mathcal{W})$. When in the possession of a training sample S we denote the resulting prior as $Q(S)$, for which $\hat{w}$ is a realisation. One of the generalization bounds we encountered under this framework was the following.

Theorem 2.1. *Let $\beta, \delta \in (0, 1)$ and $m \in \mathbb{N}$. With $\mathcal{D} \in \mathcal{M}(\mathcal{Z})$ and $\pi \in \mathcal{M}(\mathcal{W})$ we have that with probability at least $1 - \delta$ over $S \sim \mathcal{D}^m$ and for all $\rho \in \mathcal{M}(\mathcal{W})$ the following holds,*

$$R(\mathbf{w}) \leq \frac{1}{\beta}\hat{R}(\mathbf{w}) + \frac{\mathrm{KL}(\rho(S), \pi) + \log\left(\frac{1}{\delta}\right)}{2m\beta(1-\beta)}.$$

That is, the quality of our model on the unknown underlying distribution $\mathcal{D}$ can be bounded by its performance on the training sample and a term which encapsulates the similarity between our beliefs before and after the training algorithm is run. The KL-divergence term is the most significant in this bound and is something we try to minimize through the choice of prior.

2.3 Mutual Information Results

Under the information-theoretic framework, we can establish a similar bound using mutual information between the training sample and the learning algorithm. Recall, that for $\mathbf{w} \in \mathcal{W}$ and $(x, y) \sim \mathcal{D}$ we say that $l(h_{\mathbf{w}}(x), y)$ is σ-sub-Gaussian if

$$\mathbb{E}\left(e^{\lambda l(h_{\mathbf{w}}(x),y)}\right) \leq e^{\frac{\lambda^2\sigma^2}{2}}$$

for all $\lambda \in \mathbb{R}$.

Theorem 2.2. *Suppose that $l(h_{\mathbf{w}}(x), y)$ is σ-sub-Gaussian for $z \sim \mathcal{D}$ and for all $\mathbf{w} \in \mathcal{W}$, then*

$$R(\hat{\mathbf{w}}) \leq \hat{R}(\hat{\mathbf{w}}) + \sqrt{\frac{2\sigma^2 I(S; \hat{\mathbf{w}})}{m}}.$$

Note that $\hat{\mathbf{w}}$ is a random variable. It has a distribution $P_{\hat{\mathbf{w}}|S}$ that is determined by the learning algorithm and is defined on $\mathcal{W}$. This bound involves information both from the training data and the learning algorithm.

2.4 Connecting the frameworks

An approach we saw to incorporate the training data into the Bayesian framework involved holding out a subset of the training data to form a data-dependent prior. To construct this prior we suppose we have a subset S_J of S of size J and then solve the following optimization problem

$$\inf_{f: \mathcal{Z}^J \to \mathcal{M}(\mathcal{W})} \mathbb{E}(\mathrm{KL}(\rho(S), f(S_J)). \tag{1}$$

If $\hat{\mathbf{w}}$ is a random element of $\mathcal{W}$ such that $\mathbb{P}(\hat{\mathbf{w}}|S, J) = \rho(S)$ a.s then (1) is the conditional mutual information $I(S; \hat{\mathbf{w}}|S_J)$. Therefore,

$$R(\hat{\mathbf{w}}|S_J) \leq \hat{R}(\hat{\mathbf{w}}|S_J) + \sqrt{\frac{2\sigma^2 \inf_{f: \mathcal{Z}^J \to \mathcal{M}(\mathcal{W})} \mathbb{E}(\mathrm{KL}(\rho(S), f(S_J))}{m}}. \tag{2}$$

Note that the bound is not dependent on the size of S_J about S. The only influence that the size of S_J may have is in its ability to form a better prior which would in turn decrease the KL divergence. In the survey, we considered estimating the solution (1) for the case when the prior distributions were taken to be isotropic Gaussians for which their posteriors under the learning algorithm were also isotropic Gaussians. Stochastic Gradient Descent was run on the subset S_J to determine the mean of the prior, and its variance was taken to be a fixed σ_P. Refer to Figure 7 to see the value of this bound in practice.

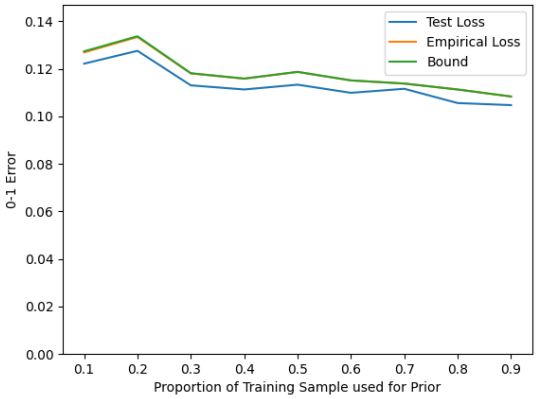

Figure 7: An evaluation of the (2) for a ReLU network with 2 hidden layers each with 600 hidden units trained to classify MNIST data.

3 KL Divergence to Measure Generalization

Here we take inspiration from the work of [3] and develop a heuristic for generalisation by considering the network's response to perturbations. In [3] Gaussian noise was added to the input and the accuracy of the outputs was then observed. They compared the performance of their networks to an idealised network. An idealised network would not suffer any performance loss from perturbing the inputs. Through this comparison, they developed two measures that were shown to be good heuristics for generalization. Their approach has the flexibility to be applied to any stage of the network's processing. For example, they were able to perturb the inputs after they had passed through some hidden layers and observe how this impacted accuracy. One of the potential drawbacks of this approach is the reliance on accuracy. Accuracy contains a relatively small amount of information about the network's performance. Therefore, it is difficult to see how it could provide a robust notion of generalization. Instead, I propose to use the KL divergence as this source of information. This is motivated by the fact that KL divergence is present in many bounds on the generalization. The intuition here is that a network that has generalised well will have robust outputs. For classification tasks, one could assume that the output distributions for the different classes are far away in some sense. To characterise the separation of these outputs distributions I will use the KL divergence. It has been empirically shown that the output distributions of neural networks are roughly Gaussian, therefore, we will approximate the output distribution using Gaussian Kernel Density Estimation (KDE). By doing this we can observe how the output distributions change as the inputs are perturbed. A network whose output distributions change drastically to small changes in the inputs shows that what it has learned is unstable. To investigate this I trained a fully connected ReLU neural network on four different datasets. I then measured the KL divergence between the network's outputs on the initial dataset and then the perturbed dataset. The dataset was perturbed by adding Gaussian noise. The variance of the Gaussian was varied from 0 to 1. The results are shown in Figure 8 and confirm the intuition. It is clear that for the datasets where the is a clear pattern the KL divergence remains low which indicates that the network has a robust representation of the data. On the other hand, for the random dataset, the KL divergence increases significantly, indicating that the learned representation of the data is unstable. An important point to note is the accuracy of each of the networks is fairly constant despite the perturbed inputs. This reinforces the fact that accuracy contains limited information and that using the KL divergence better identifies the generalization capacity of the network.

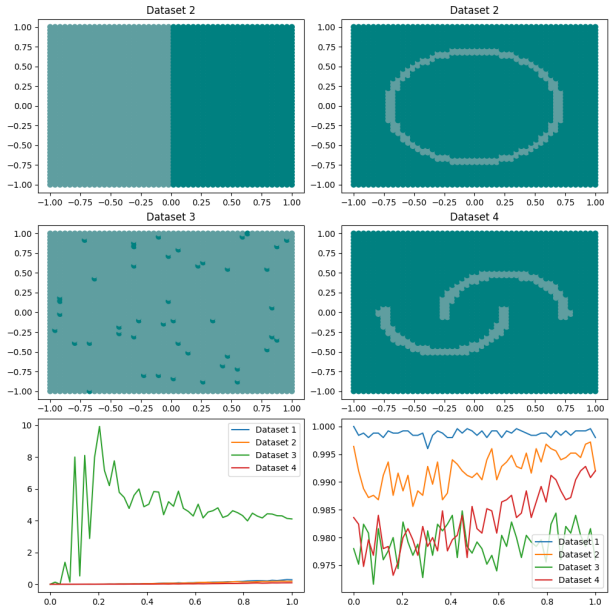

Figure 8: Using KL divergence between output distributions as a heuristic for generalization.

References

[1] Stanislav Fort, Pawel Krzysztof Nowak, and Srini Narayanan. “Stiffness: A New Perspective on Generalization in Neural Networks”. In: *CoRR* (2019).

[2] Gregory Naitzat, Andrey Zhitnikov, and Lek-Heng Lim. “Topology of deep neural networks”. In: *CoRR* (2020).

[3] Yair Schiff, Brian Quanz, Payel Das, and Pin-Yu Chen. “Gi and Pal Scores: Deep Neural Network Generalization Statistics”. In: *CoRR* (2021).