A peek inside the 'Black Box' - interpreting neural networks

Patrick Altmeyer

January, 2021

1 Motivation

Propelled by advancements in modern computer technology, deep learning has re-emerged as perhaps the most promising artificial intelligence (AI) technology of the last two decades. By treating problems as a nested, hierarchy of hidden layers deep artificial neural networks achieve the power an flexibility necessary to for AI systems to navigate complex real-world environments. Unfortunately, their very nature has earned them a reputation as *Black Box* algorithms and their lack of interpretability remains a major impediment to their more wide-spread application.

In science, research questions usually demand not just answers but also explanations and variable selection is often as important as prediction (Ish-Horowicz et al. 2019). Economists, for example, recognise the undeniable potential of deep learning, but are rightly hesitant to employ novel tools that are not fully transparent and ultimately cannot be trusted. Similarly, real-world applications of AI have come under increasing scrutiny with regulators imposing that individuals influenced by algorithms should have the right to obtain explanations (Fan, Xiong, and Wang 2020). In high-risk decision-making fields such as AI systems that drive autonomous vehicles the need for explanations is self-evident (Ish-Horowicz et al. 2019).

In light of these challenges it is not surprising that research on explainable AI has recently gained considerable momentum (Figure 1). While in this short essay we will focus on deep learning in particular, it should be noted that this growing body of literature is concerned with a broader realm of machine learning models. The rest of this note is structured as follows: section 2 provides a brief overview recent advancements towards interpreting deep neural networks largely drawing on Fan, Xiong, and Wang (2020); section 3 considers a novel entropy-based approach towards interpretability proposed by Crawford et al. (2019); finally, in section 4 we will see how this approach can be applied to deep neural networks as proposed in Ish-Horowicz et al. (2019).

2 Interpretable DL - a whistle-stop tour

Before delving further into *how* the intrinsics of deep neural networks can be disentangled we should first clarify *what* interpretability in the context of algorithms actually means . Fan, Xiong, and Wang (2020) describes model interpretability simply as the extent to which humans can "understand and reason" the model. This may concern an understanding of both the *ad-hoc* workings of the algorithm as well as the *post-hoc* interpretability of its output. In the context of linear regression, for example, *ad-hoc* workings of the model are often described through the intuitive of idea of linearly projecting the outcome variable $\mathbf{y}$ onto the column space of $\mathbf{X}$. *Post-hoc* interpretations usually center around variable importance – the main focus of the following sections. Various recent advancements tackle interpretability of DNNs from different angles depending on whether the focus is on *ad-hoc* or *post-hoc* interpretability. Fan, Xiong, and Wang (2020) further asses that model interpretability hinges on three main aspects of *simulatability*, *decomposability* and *algorithmic transparency*, but for the purpose of this short note the *ad-hoc* vs. *post-hoc* taxonomy provides a

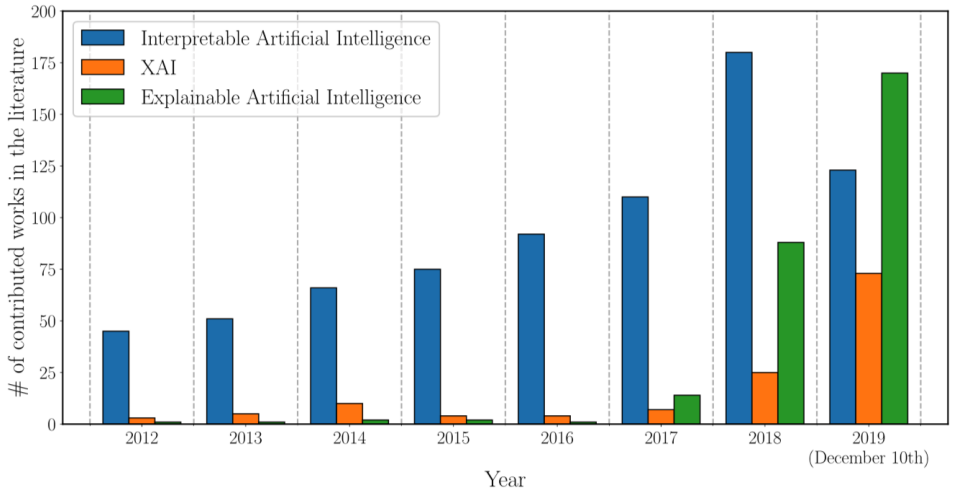

Figure 1: Rise in scientific research on explainable AI. Source: Arrieta et al. (2020)

simpler more natural framework.[1]

Understanding the *ad-hoc* intrinsic mechanisms of a DNN is inherently difficult. While generally transparency may be preserved in the presence of nonlinearity (e.g. decision trees), multiple hidden layers of networks (each of them) involving nonlinear operations are usually out of the realm of human comprehension (Fan, Xiong, and Wang 2020). Training also generally involves optimization of non-convex functions that involve an increasing number of saddle points as the dimensionality increases (Fan, Xiong, and Wang 2020). Methods to circumvent this problematic usually boil down to decreasing the overall complexity, either by regularizing the model or through proxy methods. Regularization – while traditionally done to avoid overfitting – has been found the be useful to create more interpretable representations. Monotonicity constraints, for example, impose that as the value of the a specified covariate increases model predictions either monotonically decrease or increase. Proxy methods construct simpler representations of a learned DNN, such as a rule-based decision tree. This essentially involves repeatedly querying the trained network while varying the inputs and then deriving decision rules based on the model output.

Post-hoc interpretability usually revolves around the understanding of feature importance. A greedy approach to this issue involves simply removing features one by one and checking how model predictions change. A more sophisticated approach along these lines is *Shapley* value, which draws on cooperative game theory. The Shapley value assigns varying payouts to players depending on their contribution to overall payout. In the context of neural networks input covariate $\mathbf{X}_p$ represents a player while overall payout is represented by the difference between average and individual outcome predictions.[2] Exact computations of Shapley values are prohibitive as the dimensionality increases, though approximate methods have been recently (Fan, Xiong, and Wang 2020).

The remainder of this note focuses on a novel approach to feature extraction that measures entropy shifts in a learned probabilistic neural network in response to model inputs $\mathbf{X_1}, ..., \mathbf{X}_P$. We will first introduce this methodology in the context of Gaussian process regression in the following section before finally turning to its application to Bayesian neural networks.

[1] Simulatability describes the overall, high-level understandability of the mechanisms underlying the model – put simply, the less complex the model, the higher its simulatability. Decomposability concerns the extent to which the model can be taken apart into smaller pieces – neural networks by there very nature are compositions of multiple layers. Finally, algorithmic transparency refers to the extent to which the training of the algorithm is well-understood and to some extent observable – since DNNs generally deal with optimization of non-convex functions and often lack unique solution they are inherently intransparent.

[2] For more detail see for example here

3 An entropy-based approach to variable importance

3.1 From Bayes to Gaussian Process regression

(Ish-Horowicz et al. 2019) motivate their methodology for interpreting neural networks through Gaussian Process regression. Consider the following Bayesian regression model with Gaussian priors:

$$\begin{aligned} f(\mathbf{X}|\mathbf{w}) &= \phi(\mathbf{X})^T\mathbf{w} + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \mathbf{I}) \\ \mathbf{w} &\sim \mathcal{N}(0, \frac{1}{\lambda}\mathbf{I}) \end{aligned} \tag{1}$$

This naturally gives rise to a particular example of a Gaussian Process (GP). In particular, since $\mathbf{u}(\mathbf{X}) = \Phi(\mathbf{X})^T\mathbf{w}$ is just a linear combination fo Gaussian random variables it follows a Gaussian process itself

$$\mathbf{u}(\mathbf{X}) = \Phi(\mathbf{X})^T\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}) \tag{2}$$

where $\mathbf{K}$ is the Kernel (or Gram) matrix and $K_{i,j} = k(\mathbf{X_i}, \mathbf{X_j}) = \frac{1}{\lambda}\phi(\mathbf{X_i})^T\phi(\mathbf{X_m})$ is the kernel function (Bishop 2006). In other words, the prior distribution over $\mathbf{w}$ induces a probability distribution over random functions $\mathbf{u}(\mathbf{X})$. Similarly, the GP can be understood as a prior distribution over a an infinite-dimensional reproducible kernel Hilbert space (RKHS) (Crawford et al. 2019), which in a finite-dimensional setting becomes multivariate Gaussian.

In a standard linear regression model coefficients characterize the projection of the outcome variable $\mathbf{y}$ onto the column space of the regressors $\mathbf{X}$. In particular, with ordinary least square we define:

$$\beta = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \tag{3}$$

The primary focus here is to learn the mapping from input to output. The key differentiating feature between this approach and the non-parametric model in (1) is the fact that in case of the latter we are interested in learning not only the mapping from inputs to outputs, but also the representation ($\mathbf{u}(\mathbf{X})$) of the inputs (see for example (Goodfellow, Bengio, and Courville 2016)). To be even more specific, treating the feature representation itself as random as in (1) allows us to learn non-linear relationship between the covariates $\mathbf{X}$, since they are implicitly captured by the RKHS (Crawford et al. 2019). Neural networks share this architecture and hence it is worth dwelling on it a bit further: the fact that the learned model inherently incorporates variable interactions leads to the observation that an individual feature is rarely important on its own with respect to the mapping from $\mathbf{X}$ to $\mathbf{y}$ (Ish-Horowicz et al. 2019). Hence, in order to gain an understanding of individual variable importance, one should aim to understand what role feature $\mathbf{X}_j$ plays *within* the learned model, thereby taking into account its interactions with other covariates. Formally, Crawford et al. (2019) and define the *effect size analogue* as the equivalent of the familiar regression coefficient in the non-parametric setting

$$\tilde{\beta} = \mathbf{X}^+\Phi^T\mathbf{w} = \mathbf{X}^+\mathbf{u} \tag{4}$$

where $\mathbf{X}^+ = \lim_\alpha(\mathbf{X}^T\mathbf{X} + \alpha\mathbf{I})^{-1}\mathbf{X}^T$ denotes the Moore-Penrose pseudo-inverse (see for example Goodfellow, Bengio, and Courville (2016)). Intuitively the effect size analogue can be thought of as the resulting coefficients from regressing the fitted values $\hat{\mathbf{u}}$ from the learned probabilistic model on the covariates $\mathbf{X}$. It can be interpreted in the same way as linear regression coefficients, in the sense that $\tilde{\beta}_j$ describes the marginal change in $\mathbf{u}$ given a unit increase in $\mathbf{X}_j$ holding all else constant. Note here the subtle, but crucial difference between (3) – a projection from the outcome variable onto the column space of $\mathbf{X}$ – and (4) – a projection from the learned model to $\mathbf{X}$. In other words, looking at $\tilde{\beta}$ can be thought of peeking directly into the *Block Box*. Unfortunately, as Crawford et al. (2019) point out, working with (4) is usually not straight-forward. From a practitioner's point of view, it may also not be obvious how to interpret a coefficient that describes

marginal effects of input variables on a learned model. A more useful indicator in this context would provide a measure of how much individual variables contribute to the overall variation in the learned model. For this purpose Crawford et al. (2019) propose to work with a distributional centrality measure based on $\tilde{\beta}$, which we shall turn to in the following section.

3.2 RATE

The proposed methodology in Crawford et al. (2019) and Ish-Horowicz et al. (2019) depends on the availability of a posterior distribution over $\tilde{\beta}$ in that it measures its entropic shifts in response to the introduction of covariates. The intuition is straight-forward: within the context of the learned probabilistic model is covariate $\mathbf{X_j}$ informative or not? More formally this boils down to determining if the posterior distribution of $p(\tilde{\beta}_{-j})$ is dependent on the effect of $\tilde{\beta}_j$. This can be quantified through the Kullback-Leibler divergence (KLD) between $p(\tilde{\beta}_{-j})$ and the conditional posterior $p(\tilde{\beta}_{-j}|\tilde{\beta}_j)$:

$$\text{KLD}_j = \text{KL}\left(p(\tilde{\beta}_{-j})||p(\tilde{\beta}_{-j}|\tilde{\beta}_j)\right) \tag{5}$$

Covariates that contribute significant information to the model will have KLD > 0, while for insignificant covariates KLD ≈ 0. The measure of induced entropy change gives rise to a ranking of the covariates in terms of their relative importance in the model. The RATE criterion of variable $\mathbf{X}_j$ is then simply defined as

$$\gamma_j = \frac{\text{KLD}_j}{\sum_{p=1}^{P} \text{KLD}_p} \in [0, 1] \tag{6}$$

which in light of its bounds can naturally be interpreted as $\mathbf{X}_j$'s percentage contribution to the learned model. It is worth noting that $p(\tilde{\beta}_{-j}|\tilde{\beta}_j)$ of course depends on the value of the conditioning variable. A natural choice is $\tilde{\beta}_j = 0$ which usually corresponds to the null hypothesis.

4 Application to Bayesian neural networks

In order to use the RATE criterion in the context of deep learning we need to work in the Bayesian setting. Contrary to standard artificial neural networks which work under the assumption that weights have some true latent value, Bayesian neural networks place a prior distribution over network parameters and hence treat weights as random variables (Goan and Fookes 2020). Not only does it perhaps seem more natural to treat unobserved weights as random, but the Bayesian setting also naturally gives rise to reason about uncertainty in predictions, which can ultimately help us develop more trustworthy models (Goan and Fookes 2020). A drawback of BNNs is that exact computation of posteriors is computationally challenging and often intractable (an non-trivial issue that we will turn back to in a moment).

When the prior placed over parameters is Gaussian, the output of the BNN approaches a Gaussian process as the width of the network grows, in line with the discussion in the previous section. This is exactly the assumption that Ish-Horowicz et al. (2019) work with. They propose an architecture for a multi-layer perceptron (MLP) composed of (1) an input layer collecting covariates $\mathbf{X}_1, ..., \mathbf{X}_p$, (2) a single deterministic, hidden layer and (3) an outer layer producing predictions from a probabilistic model $\mathbf{u}(\mathbf{X})$. Let $\mathbf{X}$ be a $(N \times P)$ matrix of covariates. Then formally, we have

$$\begin{aligned} \hat{\mathbf{y}} &= \sigma(\mathbf{u}) \\ \mathbf{u}(\mathbf{X}) &= \mathbf{Z}(\mathbf{X})\mathbf{w}^{(L+1)}, \quad \mathbf{w}^{(L+1)} \sim \mathcal{N}(\mathbf{m}, \mathbf{V}) \\ \mathbf{Z}(\mathbf{X}) &= f(\mathbf{X}\mathbf{w}^{(L)}) \end{aligned} \tag{7}$$

where $\sigma(.)$ is a link function and $\mathbf{u}(\mathbf{X})$ represents the probabilistic model learned in the outer layer with weights $\mathbf{w}^{(L+1)}$ assumed to be Gaussian random variables.[3] Finally, $\mathbf{Z}(\mathbf{X})$ denotes the inner (or more generally penultimate) layer, an $(N \times P)$ matrix of neural activations through $f : (\mathbf{X}\mathbf{w}^{(L)}) \mapsto \mathbf{Z}$. Figure 2 illustrates a simple representation of (7) with just one hidden layer. It should be obvious that this simple architecture can be extended to arbitrary depth and complexity, while still maintaining the high-level structure imposed by (7). This flexibility allows RATE to be applied to a wide range of Bayesian network architectures, since all that is really required is the posterior distribution over weights $\mathbf{w}^{(L+1)}$, which arises from the probabilistic outer layer. The fact that only the outer layer needs to be probabilistic has the additional benefit of mitigating the computational burden that comes with Bayesian inference, which was mentioned earlier.

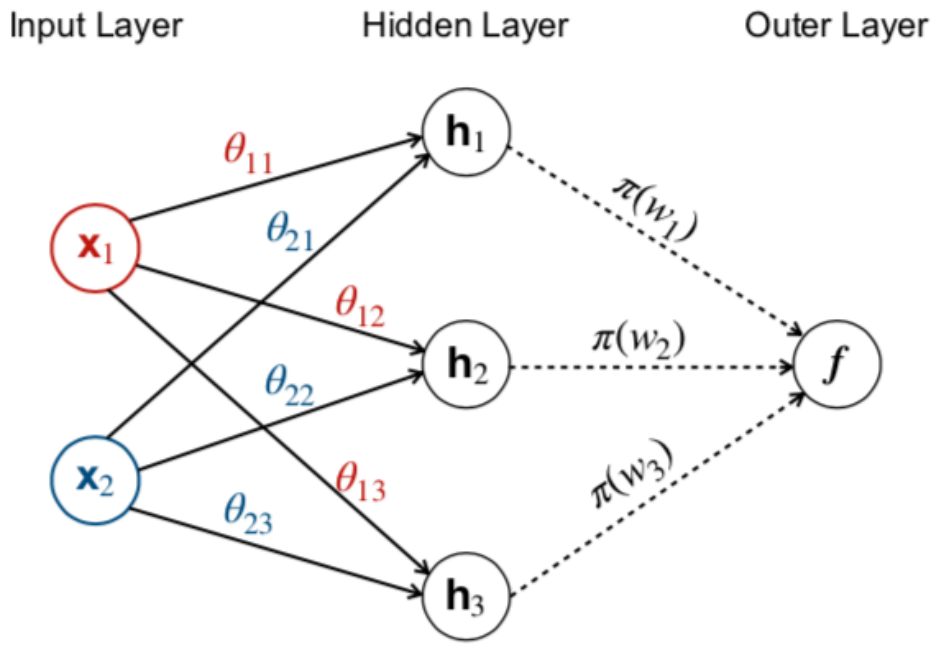

Figure 2: An MLP with a single, deterministic hidden layer and probabilistic outer layer. Source: Ish-Horowicz et al. (2019)

Having established this basic, flexible set-up the Ish-Horowicz et al. (2019) go on to derive closed-form expressions for RATE in this setting. The details are omitted here since the logic is largely analogous to what we learned in section 3, but can be found in Ish-Horowicz et al. (2019).

5 Conclusion

The RATE criterion originally proposed by Crawford et al. (2019) and shown to be applicable to Bayesian neural networks in Ish-Horowicz et al. (2019) offers an intuitive way to measure variable importance in the context of deep learning. By defining variable importance as the contribution inputs make to a probabilistic model, it implicitly incorporates the interactions between covariates and nonlinearities that the model has learned. In other words, it allows researchers to peek directly into the *Black Box*. This opens up interesting avenues for future research, as the approach can be readily applied in academic disciplines and real-world applications that rely heavily on explainability of outcomes.

[3]For simplicity I have omitted the deterministic bias term.

References

Arrieta, Alejandro Barredo, Natalia Diaz-Rodriguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador Garcia, et al. 2020. "Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges Toward Responsible AI." *Information Fusion* 58: 82–115.

Bishop, Christopher M. 2006. *Pattern Recognition and Machine Learning*. springer.

Crawford, Lorin, Seth R Flaxman, Daniel E Runcie, and Mike West. 2019. "Variable Prioritization in Nonlinear Black Box Methods: A Genetic Association Case Study." *The Annals of Applied Statistics* 13 (2): 958.

Fan, Fenglei, Jinjun Xiong, and Ge Wang. 2020. "On Interpretability of Artificial Neural Networks." *Preprint at Https://Arxiv. Org/Abs/2001.02522*.

Goan, Ethan, and Clinton Fookes. 2020. "Bayesian Neural Networks: An Introduction and Survey." In *Case Studies in Applied Bayesian Data Science*, 45–87. Springer.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press.

Ish-Horowicz, Jonathan, Dana Udwin, Seth Flaxman, Sarah Filippi, and Lorin Crawford. 2019. "Interpreting Deep Neural Networks Through Variable Importance." *arXiv Preprint arXiv:1901.09839*.

Simple MLP with single-layer

$$z_m(\mathbf{X}) = \sum_{i=1}^{k} a(x_i w_{ji}^{(1)}) \tag{8}$$

A Probabilities

Joint distribution defined as

$$p(\mathbf{w}, \mathcal{D}) = p(\mathbf{w})p(\mathcal{D}|\mathbf{w})(\#eq : joint_dist) \tag{9}$$

where for the likelihood it is common to assume that samples from $\mathcal{D}$ are i.i.d. and hence we have

$$p(\mathcal{D}|\mathbf{w}) = \prod_{i=1}^{N} \sim \mathcal{N}(\mathbf{f}(x_i), \sigma_i) \tag{10}$$

with $\mathbf{f}(x_i)$ the mean value specified by the output of the network. It is also common to assume a zero mean Gaussian with a small variance for our prior. Applying Bayes' theorem then yields

$$\pi(\mathbf{w}|\mathcal{D}) = \frac{p(\mathbf{w})p(\mathcal{D}|\mathbf{w})}{\int p(\mathbf{w})p(\mathcal{D}|\mathbf{w})d\mathbf{w}} = \frac{p(\mathbf{w})p(\mathcal{D}|\mathbf{w})}{p(\mathcal{D})} \tag{11}$$

Predictions for the outcome variable will then be an expectation fo the posterior

$$\mathbb{E}_\pi(\mathbf{f}) = \int \mathbf{f}(\mathbf{w})\pi(\mathbf{w}|\mathcal{D})d\mathbf{w} \tag{12}$$

- Prediction can then be viewed as an average of the function f weighted by the posterior. It also provides a natural setting for confidence intervals around the predictions.

B Estimating predictive moments

B.1 Variational inference

- This type of non-convex optimisation can be viewed in a Bayesian context through the lens of Variational Inference (Goan and Fookes 2020)
- VI is based on the idea to assume a suitable family of posterior distributions and then minimize the KL divergence between the true posterior and the variational posterior.
- Usually depends on imposing strong restrictions on the posterior.
- Modern approaches use SGD to optimize the variational objective.

B.2 Markov Cain Monte Carlo

- MCMC simply approximates the expectation of the posterior through repeated draws from the posterior distribution:

$$\mathbb{E}_{\pi}\left(\mathbf{f}\right) = \int \mathbf{f}(\mathbf{w})\pi(\mathbf{w}|\mathcal{D})d\mathbf{w} \approx \frac{1}{N}\sum_{i=1}^{N}\mathbf{f}(\mathbf{w}) \tag{13}$$

- MCMC enables sampling from our posterior distribution, with the samples converging to when the product of the probability density and volume are greatest (Goan and Fookes 2020)
- MCMC provides convergence to the true posterior as the number of samples approaches infinity.