Thesis

April Kopec

1 Introduction

Neural networks have become a fundamental tool in the field of machine learning. But although the lower-level implementation of a neural network is relatively simple, the behavior of neural networks has proven very difficult to analyze at a higher level. Though OpenAI has not publicly released the parameter count of GPT-4, one of the strongest currently-developed large language models, it has been estimated that that it likely has over a trillion parameters. The neural networks used in the modern day contain vast amounts of data, and use that data in a rather opaque manner; consequently, the field has struggled to understand their behavior beyond a somewhat shallow level.

One possible avenue for understanding neural networks better is to analyze the relationship between the training data they are provided and their end behavior. If the learning algorithm had been given training data which were different in some way, what predictions can we make about how the final model would have been different? *Influence functions* are one approach to this question.

1.1 Models and Influence Functions

There's a few different ways you could introduce the idea of influence functions; I've seen it presented in different ways in different papers. This way of presenting it is based on my notes, which I think were mostly based on the Anthropic paper[1].

Formally, let M be some sort of predictive model, which takes as input an independent variable $x \in \mathcal{X}$ and produces as output a predicted dependent variable $y \in \mathcal{Y}$. We could let M be any function from $\mathcal{X} \to \mathcal{Y}$, but typically we are interested in some specific class of models. For example, in linear regression we only consider models of the form $y = mx + b$. So we will say that our set of models is parameterized by a variable $\theta \in \Theta$; for the sake of being able to perform linear algebra and vector calculus, we will assume that Θ is a finite-dimensional vector space over $\mathbb{R}$, as is the case in most models used in practice. In the linear regression case, we would have $\Theta = \mathbb{R}^2$ and $\theta = (m, b)$.

So given a parameterized set of models, how do we choose one? Suppose we have a loss function $\mathcal{L}$ which tells us how poor a prediction the model with parameters θ produces for a data point z.

To choose the best model, we then find the parameters θ^* that minimizes total loss (or equivalently average loss) over our training data $\mathcal{D}$:

$$\theta^* = \operatorname*{arg\,min}_{\theta \in \Theta} \sum_{z \in \mathcal{D}} \mathcal{L}(\theta, z). \tag{1}$$

Suppose we want to determine how a particular data point $z_m \in \mathcal{D}$ influences the model parameters θ^* which we get from minimizing the loss function. One way to investigate this is to observe how θ^* changes if we change the weighting of z_m. We call the function which tells us the resulting θ^* if the weighting of a data point z_m is changed by ε the *response function*:

$$R_{z_m}(\varepsilon) = \operatorname*{arg\,min}_{\theta \in \Theta} \left(\sum_{z \in \mathcal{D}} \mathcal{L}(\theta, z_i) \right) + \varepsilon \mathcal{L}(\theta, z_m) \tag{2}$$

Note that this works whether or not z_m was already a data point in $\mathcal{D}$.

The *influence* of z_m on θ^* is the derivative of the response function r_{z_m} around $\varepsilon = 0$.

$$\mathcal{I}(z_m) = \left. \frac{\mathrm{d}R_{z_m}}{\mathrm{d}\varepsilon} \right|_{\varepsilon=0} \tag{3}$$

So the influence is a *tangent vector* to a curve in the parameter space Θ which is itself parameterized by the weight ε. To try to gauge how influential different data points are, we might look at the magnitude of their influence vector, or perhaps we might instead look at the influence of z_m on $\mathcal{L}$ or something.

1.2 Example

To help make this more concrete, let us look at a basic example: simple linear regression. Let the input space $\mathcal{X}$ and output space $\mathcal{Y}$ both be $\mathbb{R}$, and let us consider models of the form $y = mx + b$. Our data set $\mathcal{D}$ will consist of data points (x_i, y_i), and our loss function will be given by the squared error $\mathcal{L}(\theta, (x, y)) = ((\theta_1 x + \theta_0) - y)^2$.

Consider the data set $\mathcal{D} = \{(0, 0), (1, 1.5), (2, 2)\}$. The model we get with weighting each point equally is $y = 1x + 0.167$. We find that the influence of $(0, 0)$ on θ is about $(0.083, -0.13)$, of $(1, 1.5)$ is about $(0.00093, 0.11)$, and of $(2, 2)$ is about $(-0.083, 0.028)$.

This confirms some properties you might intuitively expect: for all three data points, if you imagine adjusting the model slightly to fit its influence, you will find that the model predicts the data point in question slightly better at the cost of predicting the other two points slightly worse.

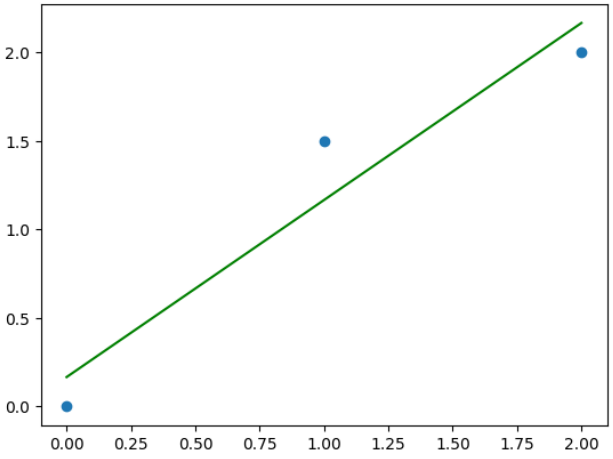

Figure 1: The three data points, along with the model using equal weights.

References

[1] Juhan Bae, Nathan Ng, Alston Lo, Marzyeh Ghassemi, and Roger Grosse. If influence functions are the answer, then what is the question?, 2022.