Sinusoidal Weights for Fourier Expressibility

Noah Schliesman

Draft: 10/06/2024

Informal Motivation

This analysis and experiment is largely inspired by the notion of Fourier decomposition. When I first learned of the subject several years ago, I became fascinated at the notion that everything in our universe could be explained as an infinite sum of waves. As is with most applications in approximation theory, I learned through my mentor and research advisor that Fourier decomposition is a beautiful way of seeing the world. While Neural Networks provide elegant approximations through hyperplane projection, I cant help but think that there might be a Fourier solution that outperforms current methods. While this experiment ultimately did not result in significant contribution, careful gradient analysis has led to a better understanding as to why.

Neural Networks and Fourier Series

Neural Networks are glorified function approximators, and in this way, it makes sense to return to the theory of Fourier series. For computational feasibility, it makes sense to begin with the discrete case:

$$S_N[n] = \sum_{k=-\infty}^{\infty} s[k] \exp\left(i2\pi \frac{k}{N} n\right) \tag{1}$$

- $S_N[n] \equiv$ discrete signal
- $s[k] \equiv$ k-th Fourier component
- $n \equiv$ discrete time index
- $\alpha = 2\pi \frac{k}{N} \equiv$ angular frequency at k
- $i \equiv$ imaginary unit

Intuitively, we can consider this notion of a Fourier activation in terms of function approximation. Ngom & Marin [1] show that Fourier networks can be effective in modeling and solving PDE's with periodic boundary conditions. Similarly, Uteluliyeva et. al [2] evaluated Fourier networks and noted that they are inferior to the simpler sigmoid and ReLU counterparts. More recently, Mehrabian et. al. [3] use a Fourier-Kolmogorov Arnold network to provide continuous and resolution-independent approximations (i.e. implicit neural representation [INR]). Furthermore, Sitzmann et. al. [4] propose a sinusoidal activation framework to represent images, wavefields, video, sound, etc., and can solve boundary problems. Clearly, extensive research encompasses functional approximation and differential analysis. Still, this notion of waves being universal approximators holds merit and forms the basis for modern physics.

Wave Functions in Neural Networks

It could be advantageous to think of the weights in a neural network as wave functions. Such a system would benefit from a bound of $[-A, A]$. Consider the following learnable parameters:

- $A \equiv$ amplitude of oscillation
- $f \equiv$ frequency of oscillation
- $\Phi \equiv$ phase shift of oscillation
- $b \equiv$ vertical offset

For weight $w \in W$, each weight becomes:

$$w = A\sin(fx + \Phi) + b \tag{2}$$

Such is conditioned for layer input X. The single neuron output y is computed as:

$$z_j = \sum_{i=1}^{n} w_i x_i + B_j = \sum_{i=1}^{n} \left(A\sin(fx_i + \Phi) + b\right) x_i + B_j \tag{3}$$

As revealed in Ba et. al.'s work on layer normalization, statistical regulation stabilizes the hidden state dynamics. Let:

$$\mu = \frac{1}{m}\sum_{j=1}^{m} z_j \tag{4}$$

$$\sigma^2 = \frac{1}{m}\sum_{j=1}^{m}(z_j - \mu)^2 \tag{5}$$

$$\hat{z_j} = \frac{z_j - \mu}{\sqrt{\sigma^2 + \epsilon}} \tag{6}$$

For numerical stability, constant $\epsilon \to 0$, and trainable parameters γ, β, we normalize the layer output:

$$y_j = \gamma\hat{z_j} + \beta_j \tag{7}$$

Gradient Analysis

To ensure stability, it is informative to perform gradient analysis:

$$\frac{\partial y}{\partial A} = \frac{\gamma x \sin(fx + \Phi)}{\sqrt{\sigma^2 + \epsilon}} \tag{8}$$

$$\frac{\partial y}{\partial f} = \frac{-Ax^2 \cos(fx + \Phi)}{\sqrt{\sigma^2 + \epsilon}} \tag{9}$$

$$\frac{\partial y}{\partial \Phi} = \frac{-Ax \cos(fx + \Phi)}{\sqrt{\sigma^2 + \epsilon}} \tag{10}$$

$$\frac{\partial y}{\partial b} = \frac{\gamma x}{\sqrt{\sigma^2 + \epsilon}} \tag{11}$$

$$\frac{\partial y}{\partial \gamma} = \frac{z_j}{\sqrt{\sigma^2 + \epsilon}} \tag{12}$$

$$\frac{\partial y}{\partial y} = \frac{\beta - \mu + x\left[A \sin(fx + \Phi) + b\right]}{\sqrt{\sigma^2 + \epsilon}} \tag{13}$$

$$\frac{\partial y}{\partial \beta} = 1 \tag{14}$$

Initially, the frequency gradient worries me. In the expected event this architecture fails, I will look to mitigate any problems with $\frac{\partial y}{\partial f}$.

Activation Function Testing

Choosing an adequate activation function is not trivial. As a baseline we test:

- (a) Sigmoid $y_{\text{act}} = \sigma(y_i)$
- (b) ReLU $y_{\text{act}} = \text{ReLU}(y_i)$

This concludes the basic architecture of our parameterizable wave function weight network, of which I have taken the liberty of calling "WFNET". Before I continue with more analysis, it is advantageous to test on a toy dataset. For this, I will create a network that harnesses the WFNET architecture to classify digits in MNIST.

Analysis of Sinusoidal Weights via Hyperbolic Attractors

In deep recurrent architectures, the phenomena of vanishing and exploding gradients pose significant challenges for training [6]. This issue can be analyzed through the lens of dynamical systems, where the recurrence relations in neural networks form discrete-time dynamical systems. By examining the system's hyperbolic attractors, we can understand gradients during backpropagation. We extend this analysis to networks with sinusoidal weights, exploring how the sinusoidal parameterization affects gradient flow and training dynamics.

Dynamical Systems Perspective

Consider a recurrent neural network (RNN) where the hidden state h_t at time step t is defined as:

$$h_t = \phi(Wh_{t-1} + Ux_t + b) \tag{15}$$

where:

- $h_t \in \mathbb{R}^n$ is the hidden state vector.
- $x_t \in \mathbb{R}^m$ is the input vector.
- $W \in \mathbb{R}^{n \times n}$ is the recurrent weight matrix.
- $U \in \mathbb{R}^{n \times m}$ is the input weight matrix.
- $b \in \mathbb{R}^n$ is the bias vector.
- $\phi(\cdot)$ is an activation function (e.g., tanh, ReLU).

The evolution of h_t can be characterized by fixed points and their stability, which are determined by the eigenvalues of the Jacobian $J_t = \frac{\partial h_t}{\partial h_{t-1}}$.

Sinusoidal Weights in Recurrence Relations

In our sinusoidal weight framework, the weights are parameterized as:

$$W_{ij} = A_{ij} \sin(f_{ij} h_{t-1,j} + \Phi_{ij}) + b_{ij} \tag{16}$$

Substituting W into the recurrence relation:

$$h_t = \phi\left(\sum_{j=1}^{n} \left[A_{ij} \sin(f_{ij} h_{t-1,j} + \Phi_{ij}) + b_{ij}\right] h_{t-1,j} + Ux_t + b\right) \tag{17}$$

This introduces a higher-order nonlinearity and dependence on h_{t-1}, complicating the dynamical analysis.

Jacobian Matrix and Gradient Propagation

The Jacobian matrix J_t is crucial for understanding gradient flow during backpropagation:

$$J_t = \frac{\partial h_t}{\partial h_{t-1}} = \phi'(z_t) \left(\frac{\partial z_t}{\partial h_{t-1}} \right) \tag{18}$$

where $z_t = Wh_{t-1} + Ux_t + b$ is the pre-activation vector, and $\phi'(z_t)$ is a diagonal matrix of activation function derivatives.

Computing the partial derivative $\frac{\partial z_t}{\partial h_{t-1}}$:

$$\frac{\partial z_{t,i}}{\partial h_{t-1,k}} = \frac{\partial}{\partial h_{t-1,k}} \left(\sum_{j=1}^{n} W_{ij} h_{t-1,j} \right) = W_{ik} + \sum_{j=1}^{n} h_{t-1,j} \frac{\partial W_{ij}}{\partial h_{t-1,k}} \tag{19}$$

Since W_{ij} depends on $h_{t-1,j}$, we need to compute $\frac{\partial W_{ij}}{\partial h_{t-1,k}}$:

$$\frac{\partial W_{ij}}{\partial h_{t-1,k}} = \delta_{jk} \left[A_{ij} f_{ij} \cos(f_{ij} h_{t-1,j} + \Phi_{ij}) \right] \tag{20}$$

where δ_{jk} is the Kronecker delta. Substituting back into $\frac{\partial z_{t,i}}{\partial h_{t-1,k}}$:

$$\frac{\partial z_{t,i}}{\partial h_{t-1,k}} = W_{ik} + h_{t-1,k} \left[A_{ik} f_{ik} \cos(f_{ik} h_{t-1,k} + \Phi_{ik}) \right] \tag{21}$$

Eigenvalue Analysis

The behavior of gradients is influenced by the eigenvalues of J_t. If the spectral radius $\rho(J_t)$ is:

- Less than 1: Gradients tend to vanish.
- Greater than 1: Gradients tend to explode.

For sinusoidal weights, W_{ij} and $\frac{\partial W_{ij}}{\partial h_{t-1,k}}$ involve sinusoidal and cosinusoidal terms, which can oscillate between $-A_{ij}$ and A_{ij}.

Hyperbolic Attractors and Stability Analysis

Hyperbolic attractors are characterized by having no eigenvalues of modulus one, leading to exponential divergence or convergence along different directions in the state space. Linearizing the system around a fixed point h^*:

$$h^* = \phi(Wh^* + Ux + b) \tag{22}$$

the Jacobian at h^* is:

$$J^* = \phi'(z^*) \left(W + \frac{\partial W}{\partial h_{t-1}} h^* \right) \tag{23}$$

The presence of $\frac{\partial W}{\partial h_{t-1}}$ introduces additional terms that can significantly affect the eigenvalues of J^*. Specifically, the term involving $\cos(f_{ij} h_{t-1,j} + \Phi_{ij})$ can cause the eigenvalues to fluctuate, potentially crossing the unit circle in the complex plane.

Conclusion

Analyzing sinusoidal weights through the perspective of hyperbolic attractors provides valuable insights into the training dynamics of neural networks with such weights. The oscillatory nature introduces complex dynamics that can exacerbate the vanishing and exploding gradient problem. Understanding these dynamics is crucial for designing effective training strategies and ensuring network stability.

References

1. M. Ngom, O. Marin, "Fourier Neural Networks as Function Approximators and Differential Equation Solvers," *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 14(2021), 641-661. https://doi.org/10.1002/sam.11331

2. U. Teululiyeva, Malka et al., "Fourier Neural Networks: A Comparative Study," *Intelligent Data Analysis*, 24.5(2020): 1107-1120.

3. A. Mehrabian et al., "Implicit Neural Representations with Fourier Kolmogorov-Arnold Networks," arXiv, 20 Sept. 2024, arXiv:2409.09323v2.

4. V. Sitzmann et al., "Implicit Neural Representations with Periodic Activation Functions," *Advances in Neural Information Processing Systems*, 33(2020): 7462-7473.

5. J. L. Ba, "Layer Normalization," arXiv preprint arXiv:1607.06450 (2016).

6. Y. Bengio, P. Simard, and P. Frasconi, "Learning Long-Term Dependencies with Gradient Descent is Difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.

7. G. Klambauer et al., "Self-Normalizing Neural Networks," *Advances in Neural Information Processing Systems*, 30, 2017.