Complex Valued Neural Networks

Reviewed by Eun Woo Im

Introduction

- In many practical applications, complex numbers are often used such as in telecommunications, robotics, bioinformatics, speech recognition, etc.

- This suggests that ANNs using complex numbers to represent inputs, outputs, and parameters.

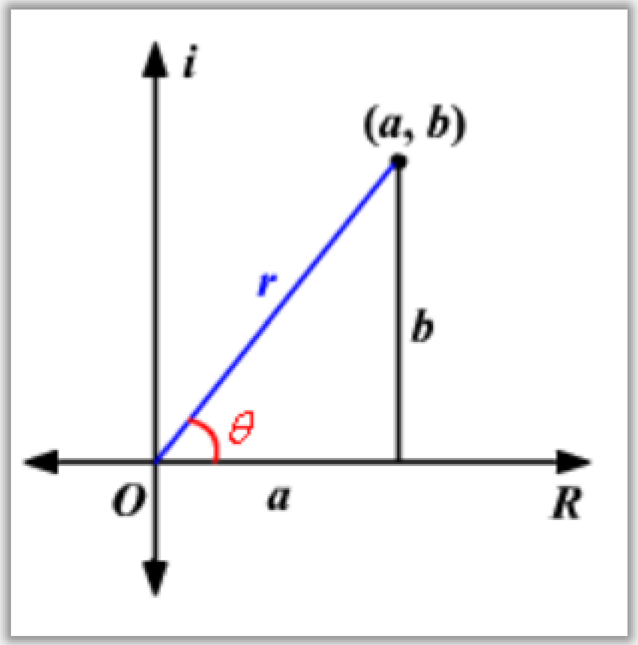
- Multiplication function which results in a phase rotation and amplitude modulation yields an advantageous reduction of the degree of freedom.

Complex Numbers

Basic Formula for Complex Numbers

$$z = a + bi$$

Real Imaginary



$$z = r(\cos\theta + i\sin\theta)$$

$$e^{\theta i} = \cos\theta + i\sin\theta$$

Derived from Taylor Expansion

Cartesian Coordinate

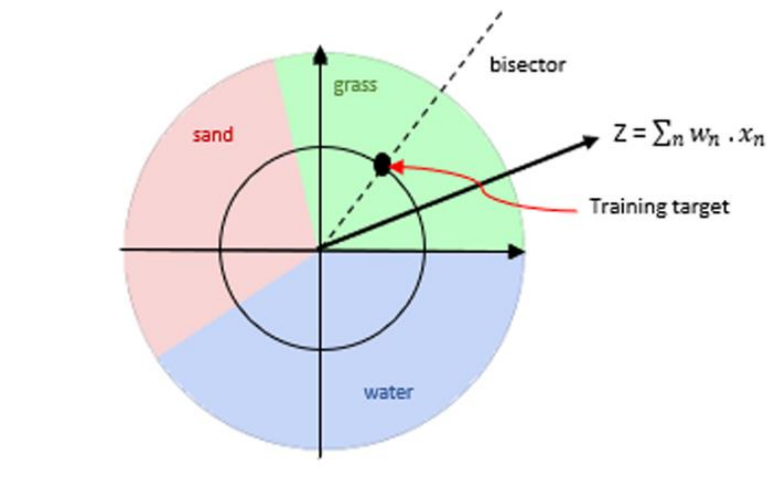
Polar Coordinate

Euler Coordinate

Activation Functions

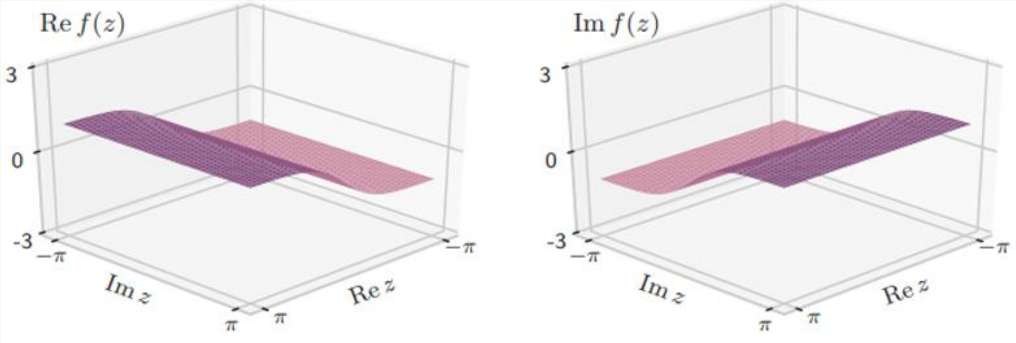
Requirement: Nonlinear, susceptible to gradient exploding/vanishing

MVN: $\boldsymbol{z = (w_0 + w_1 x_1 + \cdots + w_n x_n)}$



1. **Multi-valued Neuron, neural element with n inputs and one output lying on the unit circle, with complex-valued weight**

   → all outputs of the function are the k-th roots of unity
   $f(z) = \exp\left(\frac{i2\pi j}{k}\right) if\ 2\pi j/k \leq \arg(z) \leq 2\pi(j+1)/kq$
   can extended to continuous valued inputs by $k \to \infty$



2. **Fully complex networks**

   Differentiable functions.
   For example, hyperbolic tangent activation function.

Activation Functions

3. Radial Basis Function

Kernel trick for mapping into higher dimension.
$z = (x_1, \dots, x_n), K_{RBF} = \exp\left(-\frac{1}{2}\left(x_i - x_j\right)^2\right)$

4. Non-parametric functions

For example, ReLU function that separately on both real and imaginary parts.

Tradeoff: boundness vs differentiable

Activation Functions

activation	$f(z) =$
ETFs (T. Kim & Adalı, 2003)	see Table 3.1
Georgiou and Koutsougeras (1992)[a]	$\frac{z}{c+\frac{1}{r}\lvert z\rvert}$
Hirose (1992a)[b]	$\tanh\left(\frac{\lvert z\rvert}{m}\right)e^{\mathrm{i}\arg z}$
Type A (Kuroe et al., 2003)[c]	$f^{(r)}(\mathrm{Re}\,z) + \mathrm{i}f^{(i)}(\mathrm{Im}\,z)$
Type B (Kuroe et al., 2003)[d]	$\psi(\lvert z\rvert)e^{\mathrm{i}\varphi(\arg z)}$
modReLU (Arjovsky et al., 2016)[e]	$\mathrm{ReLU}(\lvert z\rvert + b)e^{\mathrm{i}\arg z}$
zReLU (Guberman, 2016)	$\begin{cases} z & \text{if} \quad \arg z \in [0, \pi/2] \\ 0 & \text{otherwise} \end{cases}$
$\mathbb{C}$ReLU (Trabelsi et al., 2017)	$\mathrm{ReLU}(\mathrm{Re}\,z) + \mathrm{i}\,\mathrm{ReLU}(\mathrm{Im}\,z)$

[a] c and r are constants.
[b] m is a constant.
[c] $f^{(r)}$ and $f^{(i)}$ are nonlinear real functions.
[d] ψ and φ are nonlinear non-negative real functions.
[e] b is a trainable bias parameter.

Table 3.1: Elementary transcendental functions and their derivatives.

$f(z)$	$\frac{df}{dz}$
Circular	
$\tan z = \frac{e^{iz} - e^{-iz}}{i(e^{iz} + e^{-iz})}$	$\frac{d}{dz}\tan z = \sec^2(z)$
$\sin z = \frac{e^{iz} - e^{-iz}}{2i}$	$\frac{d}{dz}\sin z = \cos z$
Inverse circular	
$\arctan z = \frac{1}{2}i[\ln(1 - iz) - \ln(1 + iz)]$	$\frac{d}{dz}\arctan z = \frac{1}{1+z^2} : z \neq \pm i$
$\arcsin z = -i\ln\left(iz + \sqrt{1 - z^2}\right)$	$\frac{d}{dz}\arcsin z = \frac{1}{\sqrt{1-z^2}} : z \neq \pm 1$
$\arccos z = \frac{1}{2}\pi + i\ln\left(iz + \sqrt{1 - z^2}\right)$	$\frac{d}{dz}\arccos z = -\frac{1}{\sqrt{1-z^2}} : z \neq \pm 1$
Hyperbolic	
$\mathrm{arctanh}\, z = \frac{1}{2}[\ln(1 + z) - \ln(1 - z)]$	$\frac{d}{dz}\mathrm{arctanh}\, z = \frac{1}{1-z^2} : z \neq \pm 1$
$\mathrm{arcsinh}\, z = \ln\left(z + \sqrt{1 + z^2}\right)$	$\frac{d}{dz}\mathrm{arcsinh}\, z = \frac{1}{\sqrt{1+z^2}} : z \neq \pm 1$
Inverse hyperbolic	
$\mathrm{arctanh}\, z = \frac{1}{2}[\ln(1 + z) - \ln(1 - z)]$	$\frac{d}{dz}\mathrm{arctanh}\, z = \frac{1}{1-z^2} : z \neq \pm 1$
$\mathrm{arcsinh}\, z = \ln\left(z + \sqrt{1 + z^2}\right)$	$\frac{d}{dz}\mathrm{arcsinh}\, z = \frac{1}{\sqrt{1+z^2}} : z \neq \pm 1$

Optimization and Learning

Gradient-based Approach

Note: All activations should be initially assumed to exist for all neuron outputs so that Cauchy-Riemann Equations are satisfied.

$$f(z) = u(z) + i \cdot v(z) \quad z = x + iy.$$

$$\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y} \qquad \frac{\partial u}{\partial y} = -\frac{\partial v}{\partial x}$$

The process of learning with complex domain back-propagation is similar to the process in the real domain. Following is used for complex square mean.

$$\mathcal{L}(e) \quad = \sum_{k=0}^{N-1} e_k \bar{e}_k. \quad \mathcal{L}(e_{log}) = \frac{1}{2}\left(log\left[\frac{\hat{r}_k}{r_k}\right]^2 + [\hat{\phi}_k - \phi_k]^2 \right)$$

$$(e_{log}) := \sum_{k=0}^{N-1} (\log\ (o_k) - \log(d_k))\,\overline{(\log\ (o_k) - \log(d_k))}$$

Optimization and Learning

Non-gradient-based Approach

For a single neuron, weight correction in MVN is determined by the neuron's error, and learning is reduced to a simple movement along the unit circle.

$$\begin{aligned} \tilde{w}_i^{kj} &= w_i^{kj} + \frac{C_{kj}}{(N_{j-1}+1)}\epsilon_{kj}\bar{\tilde{Y}}_{i,j-1}, \quad i = 1, \ldots, n \\ \tilde{w}_0^{kj} &= w_0^{kj} + \frac{C_{kj}}{(N_{j-1}+1)}\epsilon_{kj} \end{aligned}$$

$$\begin{aligned} \tilde{w}_i^{k1} &= w_i^{k1} + \frac{C_{k1}}{(n+1)}\epsilon_{k1}\bar{x}_i, \quad i = 1, \ldots, n \\ \tilde{w}_0^{k1} &= w_0^{k1} + \frac{C_{k1}}{(n+1)}\epsilon_{kj} \end{aligned}$$

Pros

1. Easy to implement
2. No problem from gradient
   e.g., local minima
3. Possible to build hybrid networks

Input and Output Representations

- **Input can be complex either naturally or by design**

  e.g., Fourier transform on images, radio frequency

- **However, real-values are more appropriate if the goal is to perform inference over a probability distribution of complex parameters.**

- **Toy examples trained to learn a function in 4 ways.**

1) amplitude-phase where a real-valued vector is formed from concatenating the phase offset and amplitude parameters;
2) complex representation where the phase offset is used as the phase of the complex phasor;
3) real-imaginary representation where the real and imaginary components of the complex vector in 2) are used as real vectors;
4) augmented complex representation which is a concatenation of the complex vector and its conjugate.

Input and Output Representations

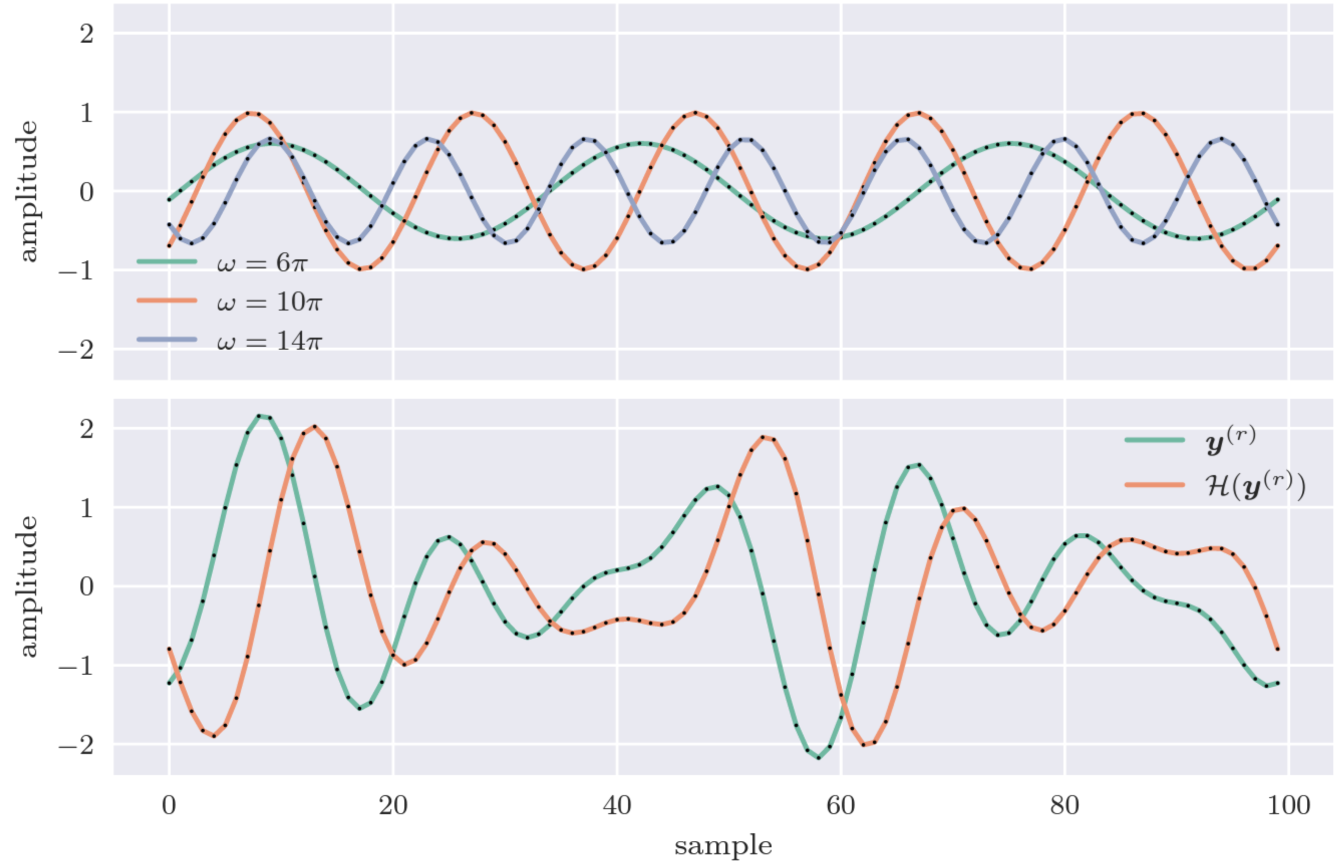


Figure 3.5: Top: three real sinusoids corresponding to arbitrary vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\varphi}$. Bottom: corresponding real-valued output $\boldsymbol{f}(\boldsymbol{\alpha}, \boldsymbol{\varphi})$ and the Hilbert transform of $\boldsymbol{f}$. Black markers indicate sample points corresponding to $\boldsymbol{t}$.

$$\boldsymbol{f}(\boldsymbol{\alpha}, \boldsymbol{\varphi}) = \sum_{i \in \{0,1,2\}} \alpha_i \sin(\omega_i \boldsymbol{t} + \varphi_i), \quad \text{given}$$

$$\boldsymbol{\omega} = \begin{bmatrix} 6\pi & 10\pi & 14\pi \end{bmatrix}^{\mathsf{T}} \quad \text{and}$$

$$\boldsymbol{t} = \begin{bmatrix} 0 & \frac{1}{100} & \cdots & \frac{99}{100} \end{bmatrix},$$

Andy Sarroff, Complex Neural Networks for Audio, Dartmouth Digital Commons 2018

Input and Output Representations

Amplitude-Phase. Arguably the simplest input representation is to simply concatenate the amplitude and phase offset parameters into a real-valued vector,

$$\boldsymbol{x}^{(ap)} := \begin{bmatrix} \cdots & \alpha_i & \varphi_i & \cdots \end{bmatrix}^\mathsf{T} \in \mathbb{R}^{2K}.$$

Complex. Alternatively, the input parameters can be composed into complex numbers by letting the phase offset represent the phase of a complex phasor.

$$\boldsymbol{x}^{(c)} := \begin{bmatrix} \cdots & \alpha_i \mathrm{e}^{\mathrm{i}\varphi_i} & \cdots \end{bmatrix}^\mathsf{T} \in \mathbb{C}^{K}.$$

The vector $\boldsymbol{x}^{(c)}$ is complex by design, where we use the phase of a complex number to represent a parameter that represents rotation or angle.

Real-Imaginary. The complex vector in Eq. (3.61) could also be broken into its real and imaginary parts to form a real vector.

$$\begin{aligned} \boldsymbol{x}^{(ri)} &:= \left[\mathrm{Re}\left\{\boldsymbol{x}^{(c)}\right\}^\mathsf{T} \quad \mathrm{Im}\left\{\boldsymbol{x}^{(c)}\right\}^\mathsf{T} \right]^\mathsf{T} \\ &= \begin{bmatrix} \cdots & \alpha_i \cos\varphi_i & \cdots & a_0 \sin\varphi_0 & \cdots \end{bmatrix}^\mathsf{T} \in \mathbb{R}^{2K}. \end{aligned}$$

Augmented Complex. Finally, we may use the augmented complex vector, which is the complex vector concatenated with its conjugate.

$$\begin{aligned} \boldsymbol{x}^{(ac)} &:= \left[\left(\boldsymbol{x}^{(c)}\right)^\mathsf{T} \quad \left(\boldsymbol{x}^{(c)}\right)^* \right]^\mathsf{T} \in \mathbb{C}^{2K}, \\ &= \begin{bmatrix} \cdots & \alpha_i \mathrm{e}^{\mathrm{i}\varphi_i} & \cdots & \alpha_i \mathrm{e}^{-\mathrm{i}\varphi_i} & \cdots \end{bmatrix}^\mathsf{T} \end{aligned} \tag{3.65}$$

Andy Sarroff, Complex Neural Networks for Audio, Dartmouth Digital Commons 2018

Challenges and Potential Research

1. Limitation that the complex-valued activation is not complex-differentiable and bounded at the same time.

2. Not many deep learning libraries are optimized for complex-valued operations.

3. Alternative methods of complex-valued weight initialization for CVNNs.

4. Number of computational complexity increase. Network may gain more expressiveness but run the risk of overfitting due to the increase in parameters as the network goes deeper.

Andy Sarroff, Complex Neural Networks for Audio, Dartmouth Digital Commons 2018