Gradient of Complex Neural Network

tomas.ukkonen@novelinsight.fi, 2020

Complex valued neural networks extend computing capacity of neural networks. However, analytical calculation of the gradient of the network's error terms cannot be done using standard complex differentation and requires Wirtinger-calculus which defines function $\boldsymbol{f}(\boldsymbol{z}, \bar{\boldsymbol{z}})$ which is not fully holomorphic. Good reference paper about Wirtinger calculus is *The Complex Gradient Operator and the CR-Calculus. Ken Kreutz-Delgado. 2009. (arXiv.org 0906.4835v1).*

Consider a two-layer complex valued neural network

$$\boldsymbol{y}(\boldsymbol{z}) = \boldsymbol{f}(\boldsymbol{W}^{(2)}\, \boldsymbol{g}(\boldsymbol{W}^{(1)}\boldsymbol{z} + \boldsymbol{b}^{(1)}) + \boldsymbol{b}^{(2)}).$$

We have linear algebra opetations $\boldsymbol{v}^{(1)} = \boldsymbol{W}^{(1)}\boldsymbol{z} + \boldsymbol{b}^{(1)}$ which satisfy Cauchy-Riemann conditions and which can be derivated using standard complex analysis. But additionally to this we have non-linearities which typically don't satisfy Cauchy-Riemann conditions. The standard non-linearities like leaky ReLU (rectifier $y = \max(0.01x, x)$) don't satisfy Cauchy-Riemann conditions.

For simplicity we want to have non-linearity which satisfies Cauchy-Riemann conditions so that we can have analytical derivate of the function (NOT REALLY NEEDED ReLU works ok).

Function $f(z) = e^z$ satisfies Cauchy-Riemann conditions which can be seen by calculating partial derivates. Because we cannot use ReLU we can use *softplus* non-linearity which can be derivated using standard complex calculus.

$$f(z) = \ln(1 + e^{kz})/k, k = 1.5$$
$$f'(z) = (1 + e^{-kz})^{-1}$$

We can extend ReLU to complex numbers by defining

$$f(a + b\,i) = \max(0.1\,a, a) + \max(0.1\,b, b)i$$

$f'(z) = f(z)/z$, since $f(z) = c\,z$ is essentially (part-wise) linear function.

After these definitions we can calculate gradient of the neural network $\boldsymbol{y}(\boldsymbol{z})$ function using simple standard complex calculus. But we need to additionally calculate derivate of error function

$$f(\boldsymbol{w}, \bar{\boldsymbol{w}}) = \tfrac{1}{2}E\{\boldsymbol{z}(\boldsymbol{w})^H\boldsymbol{z}(\boldsymbol{w})\} = \tfrac{1}{2}E\{(\boldsymbol{y}(\boldsymbol{x}|\boldsymbol{w}) - \boldsymbol{y})^H(\boldsymbol{y}(\boldsymbol{x}|\boldsymbol{w}) - \boldsymbol{y})\}.$$

We can calculate partial derivates

$$\frac{\partial f}{\partial \boldsymbol{w}} = \frac{1}{2}\left(\frac{\partial \frac{1}{2}\|\boldsymbol{z}(\boldsymbol{w})\|^2}{\partial \boldsymbol{z}}\frac{\partial \boldsymbol{z}(\boldsymbol{w})}{\partial \boldsymbol{w}} + \frac{\partial \frac{1}{2}\|\boldsymbol{z}(\boldsymbol{w})\|^2}{\partial \bar{\boldsymbol{z}}}\frac{\partial \bar{\boldsymbol{z}}(\boldsymbol{w})}{\partial \boldsymbol{w}}\right)$$

that $f()$ is real valued so we can calculate conjugate gradient by using the result: $\frac{\partial f}{\partial \bar{\boldsymbol{w}}} = = \overline{\frac{\partial f}{\partial \boldsymbol{w}}}$.

Additionally because neural network $\boldsymbol{z}(\boldsymbol{w}) = \boldsymbol{y}(\boldsymbol{z}|\boldsymbol{w})$ satisfies Cauchy-Riemann conditions this means $\frac{\partial \bar{\boldsymbol{z}}(\boldsymbol{w})}{\partial \boldsymbol{w}} = \left(\overline{\frac{\partial \boldsymbol{z}(\boldsymbol{w})}{\partial \bar{\boldsymbol{w}}}}\right) = \boldsymbol{0}$. Also the partial derivate $\frac{\partial \|\boldsymbol{z}(\boldsymbol{w})\|^2}{\partial \boldsymbol{z}} = \frac{\partial(\boldsymbol{z}(\boldsymbol{w})^H\boldsymbol{z}(\boldsymbol{w}))}{\partial \boldsymbol{z}} = \boldsymbol{z}(\boldsymbol{w})^H$. This means we have derivates

$$\frac{\partial f(\boldsymbol{w}, \bar{\boldsymbol{w}})}{\partial \boldsymbol{w}} = \frac{1}{2}\boldsymbol{z}(\boldsymbol{w})^H\frac{\partial \boldsymbol{z}(\boldsymbol{w})}{\partial \boldsymbol{w}}$$
$$\frac{\partial f(\boldsymbol{w}, \bar{\boldsymbol{w}})}{\partial \bar{\boldsymbol{w}}} = = \overline{\frac{\partial f}{\partial \boldsymbol{w}}} = \frac{1}{2}\boldsymbol{z}(\boldsymbol{w})^T\overline{\frac{\partial \boldsymbol{z}(\boldsymbol{w})}{\partial \boldsymbol{w}}}$$

Now we want to transform these to real valued coordinate system $r = (\boldsymbol{x}_k, y_k) \sim \mathbb{R}^2$ where real and imaginary values are separated.

This can be done using linear mapping $\boldsymbol{r} = \frac{1}{2}\boldsymbol{J}^H \boldsymbol{c}$, $\boldsymbol{c} = \begin{bmatrix} \begin{pmatrix} \boldsymbol{z} \\ \bar{\boldsymbol{z}} \end{pmatrix} \end{bmatrix}$ and $\frac{\partial f}{\partial \boldsymbol{r}} = \boldsymbol{J}^T \frac{\partial f}{\partial \boldsymbol{c}}$ where $\boldsymbol{J}^T = \begin{pmatrix} I & I \\ Ij & -Ij \end{pmatrix}$ so we get results ($\boldsymbol{w} = \boldsymbol{x} + \boldsymbol{y}\,j$):

$$\frac{\partial \operatorname{Re}(f(\boldsymbol{w}))}{\partial \boldsymbol{x}} = \frac{1}{2}\Big(\boldsymbol{z}(\boldsymbol{w})^H \frac{\partial \boldsymbol{z}(\boldsymbol{w})}{\partial \boldsymbol{w}} + \boldsymbol{z}(\boldsymbol{w})^T \overline{\frac{\partial \boldsymbol{z}(\boldsymbol{w})}{\partial \boldsymbol{w}}} \Big)$$

$$\frac{\partial \operatorname{Im}(f(\boldsymbol{w}))}{\partial \boldsymbol{y}} = \frac{1}{2}\Big(\boldsymbol{z}(\boldsymbol{w})^H \frac{\partial \boldsymbol{z}(\boldsymbol{w})}{\partial \boldsymbol{w}} - \boldsymbol{z}(\boldsymbol{w})^T \overline{\frac{\partial \boldsymbol{z}(\boldsymbol{w})}{\partial \boldsymbol{w}}} \Big) j$$

Plugging these equations into formula $f(\boldsymbol{w}) = \boldsymbol{x} + \boldsymbol{y}\,j = \operatorname{Re}(f(\boldsymbol{w})) + \operatorname{Im}(f(\boldsymbol{w}))j$ gives

$$\frac{\partial \boldsymbol{f}(\boldsymbol{w})}{\partial \boldsymbol{w}} = \frac{1}{2}\Big(\boldsymbol{z}(\boldsymbol{w})^H \frac{\partial \boldsymbol{z}(\boldsymbol{w})}{\partial \boldsymbol{w}} + \boldsymbol{z}(\boldsymbol{w})^T \overline{\frac{\partial \boldsymbol{z}(\boldsymbol{w})}{\partial \boldsymbol{w}}} \Big) - \frac{1}{2}\Big(\boldsymbol{z}(\boldsymbol{w})^H \frac{\partial \boldsymbol{z}(\boldsymbol{w})}{\partial \boldsymbol{w}} - \boldsymbol{z}(\boldsymbol{w})^T \overline{\frac{\partial \boldsymbol{z}(\boldsymbol{w})}{\partial \boldsymbol{w}}} \Big)$$

$$\frac{\partial \boldsymbol{f}(\boldsymbol{w})}{\partial \boldsymbol{w}} = \boldsymbol{z}(\boldsymbol{w})^T \overline{\frac{\partial \boldsymbol{z}(\boldsymbol{w})}{\partial \boldsymbol{w}}}$$

In practice optimization seem to generate large weight values $\boldsymbol{w}$ which cause floating point errors. This means we need to add regularizer to our optimized function.

$$f(\boldsymbol{w}, \bar{\boldsymbol{w}}) = \frac{1}{2} E\{\boldsymbol{z}(\boldsymbol{w})^H \boldsymbol{z}(\boldsymbol{w})\} + \frac{1}{2}\alpha \|\boldsymbol{w}\|^2 = \frac{1}{2} E\{(\boldsymbol{y}(\boldsymbol{x}|\boldsymbol{w}) - \boldsymbol{y})^H (\boldsymbol{y}(\boldsymbol{x}|\boldsymbol{w}) - \boldsymbol{y})\} + \frac{1}{2}\alpha \|\boldsymbol{w}\|^2.$$

This means

$$\frac{\partial f(\boldsymbol{w}, \bar{\boldsymbol{w}})}{\partial \boldsymbol{w}} = \frac{1}{2}\boldsymbol{z}(\boldsymbol{w})^H \frac{\partial \boldsymbol{z}(\boldsymbol{w})}{\partial \boldsymbol{w}} + \frac{1}{2}\,\alpha \bar{\boldsymbol{w}}$$

$$\frac{\partial f(\boldsymbol{w}, \bar{\boldsymbol{w}})}{\partial \bar{\boldsymbol{w}}} = = \overline{\frac{\partial f}{\partial \boldsymbol{w}}} = \frac{1}{2}\boldsymbol{z}(\boldsymbol{w})^T \overline{\frac{\partial \boldsymbol{z}(\boldsymbol{w})}{\partial \boldsymbol{w}}} + \frac{1}{2}\,\alpha\, \boldsymbol{w}$$

And real/imaginary parts are:

$$\frac{\partial \operatorname{Re}(f(\boldsymbol{w}))}{\partial \boldsymbol{x}} = \frac{1}{2}\Big(\boldsymbol{z}(\boldsymbol{w})^H \frac{\partial \boldsymbol{z}(\boldsymbol{w})}{\partial \boldsymbol{w}} + \boldsymbol{z}(\boldsymbol{w})^T \overline{\frac{\partial \boldsymbol{z}(\boldsymbol{w})}{\partial \boldsymbol{w}}} \Big) + \boldsymbol{\alpha} \operatorname{Re}(\boldsymbol{w})$$

$$\frac{\partial \operatorname{Im}(f(\boldsymbol{w}))}{\partial \boldsymbol{y}} = \frac{1}{2}\Big(\boldsymbol{z}(\boldsymbol{w})^H \frac{\partial \boldsymbol{z}(\boldsymbol{w})}{\partial \boldsymbol{w}} - \boldsymbol{z}(\boldsymbol{w})^T \overline{\frac{\partial \boldsymbol{z}(\boldsymbol{w})}{\partial \boldsymbol{w}}} \Big) j + \boldsymbol{\alpha} \operatorname{Im}(\boldsymbol{w})$$

$$\frac{\partial \boldsymbol{f}(\boldsymbol{w})}{\partial \boldsymbol{w}} = \boldsymbol{z}(\boldsymbol{w})^T \overline{\frac{\partial \boldsymbol{z}(\boldsymbol{w})}{\partial \boldsymbol{w}}} + \boldsymbol{\alpha}\, \boldsymbol{w}$$

Adding regularizer term with $\alpha = 10^{-6}$ solves the problem (weight vectors are initially $N(0, I)$ / sqrt(dim(w)) or something so $N(0, I)$ input data should initially have something like unit variance and zero mean when going through the linear network.

Although this neural network model uses complex numbers it fulfills Cauchy-Riemann conditions so it restricts the possible complex number functions. Improvement would use non standard complex derivable nonlinearities like extension of LeakyRELU extended complex numbers. After testing quality of found solutions test how TensorFlow's implementation of complex numbers with non-standard functions (automatic derivation) performs.

TODO: Calculate also gradient of norm $\|\boldsymbol{f}(\boldsymbol{z}) - \boldsymbol{y}\|$ instead of squared error.