Generalized Artifical Neural Networks

Approximated by Weierstrass Polynomials

ANON

February 5, 2016

1 Functional Neural Networks

1.1 Derivation

Now that the functional neural network has been theoretically defined, the next step is to determine whether the implementation of a numerical algorithm is possible. The problem is approached using techniques already developed for discretized networks. To produce an output for a network with L layers, the implementation would propagate through all the layers while caching previous computations along the way. Likewise, for the error back-propagation method, we will cache and eliminate variables until we are able to define an algorithm that is suitably computable.

In order to begin exploration of such an implementation, we need to both more rigorously define our notion of the weight surface and what is itself computationally evaluable.

As were for discretized neural networks, the parameters of functional neural networks are defined. In particular, the weight surface for each layer is ideally optimized for a desired output operator. To construct these weight surfaces we will parameterize them using coefficients of Weierstrass polynomials. Consider the following definition.

Definition 1.1.1. *We say that a weight function is parameterized if it is defined as follows.*

$$w^{(l)}(i,j) = \sum_{x_{2l+1}}^{Z_Y^{(l)}} \sum_{x_{2l}}^{Z_X^{(l)}} k^{(l)}_{x_{2l},x_{2l+1}} j_l^{x_{2l}} j_{l+1}^{x_{2l+1}} \tag{1.1.1}$$

where $k_{a,b}$ is some real valued coefficient and the order of the polynomial is arbitrary.

With that in mind, let us define the notion of numerical integrability (and thereby evaluability).

Definition 1.1.2. *Let $f : \mathbb{R}^{2n} \to \mathbb{R}$, We say that some function ϕ of the form*

$$\phi(x) = \int_{E \subset \mathbb{R}^n} f(x,y)dy \tag{1.1.2}$$

can be numerical integrated if and only if

$$\phi(x) = h(x) \int_{E \subset \mathbb{R}^n} g(y)dy \tag{1.1.3}$$

for some $h, g : \mathbb{R}^n \to \mathbb{R}$.

Using the above definition we will explore the feed-forward and back-propagation actions of any given $\mathcal{F}$.

1.1.1 Feed-Forward Propagation

Consider the notion of numerical integrability defined in definition 1.1.2 applied to the calculation of output for some functional neural network.

Theorem 1.1.3. *If $\mathcal{F}$ is a functional neural network with L consecutive layers, then given any l such that $0 \leq l < L$, $\sigma^{(l+1)}$ is numerically integrable, and if ξ is any continuous and Riemann integrable input function, then $\mathcal{F}[\xi]$ is numerically integrable.*

Proof. Consider the first layer. We can write the sigmoidal output of the $(l+1)^{\text{th}}$ layer as a function of the previous layer; that is,

$$\sigma^{(l+1)} = g\left(\int_{R^{(l)}} w^{(l)}(j_l, j_{l+1})\sigma^{(l)}(j_l)\, dj_l\right). \tag{1.1.4}$$

Clearly this composition can be expanded using the polynomial definition of the weight surface. Hence

$$\begin{aligned}\sigma^{(l+1)} &= g\left(\int_{R^{(l)}} \sigma^{(l)}(j_l) \sum_{x_{2l+1}}^{Z_Y^{(l)}} \sum_{x_{2l}}^{Z_X^{(l)}} k_{x_{2l},x_{2l+1}} j_l^{x_{2l}} j_{l+1}^{x_{2l+1}}\, dj_l\right) \\ &= g\left(\sum_{x_{2l+1}}^{Z_Y^{(l)}} j^{x_{2l+1}} \sum_{x_{2l}}^{Z_X^{(l)}} k_{x_{2l},x_{2l+1}} \int_{R^{(l)}} \sigma^{(l)}(j_l) j_l^{x_{2l}}\, dj_l\right),\end{aligned} \tag{1.1.5}$$

and therefore $\sigma^{(l+1)}$ is numerically integrable. For the purpose of constructing an algorithm, let $I_{x_{2l}}^{(l)}$ be the evaluation of the integral in the above definition for any given x_{2l}

It is important to note that the previous proof requires that $\sigma^{(l)}$ be Riemann integrable. Hence, with ξ satisfying those conditions it follows that every $\sigma^{(l)}$ is integrable inductively. That is, because $\sigma^{(0)}$ is integrable it follows that by the numerical integrability of all l, $\mathcal{F}[\xi] = \sigma^{(L)}$ is numerically integrable. This completes the proof. □

Using the logic of the previous proof, it follows that the development of some inductive algorithm is possible.

1.1.2 Feed-Forward Algorithm

Fortunately in the proof it was shown that by calculation of a constant, I, on each layer, the functional neural network becomes numerically integrable. The mechanism by which this can occur leads us to a simple algorithm for calculating $\mathcal{F}[\xi]$:

1. For each $l \in \{0, \ldots, L-1\}$
   (a) For all $t \in Z_X^{(l)}$, calculate
   $$I_t^{(l)} = \int_{R^{(l)}} \sigma^{(l)}(j_l) j_l^t\, dj_l. \tag{1.1.6}$$
   (b) Calculate, for every $s \in Z_Y^{(l)}$,
   $$C_s^{(l)} = \sum_{x_{2l}}^{Z_X^{(l)}} k_{x_{2l},s} I_{x_{2l}}^{(l)} \tag{1.1.7}$$
   (c) Finally, using (1.1.6) and (1.1.7), cache
   $$\sigma^{(l+1)} = g\left(\sum_{x_{2l+1}}^{Z_Y^{(l)}} j^{x_{2l+1}} C_{x_{2l+1}}^{(l)}\right) \tag{1.1.8}$$
   for use in the next iteration of loop.
2. The last $\sigma^{(l)}$ calculated is the output of the functional neural network.

1.1.3 Continuous Error Backpropagation

Just as important as the feed-forward of neural network algorithms is the notion of training. As is common with many non-convex problems with discretized neural networks, a stochastic gradient descent method will be developed using a continuous analogue to error backpropagation.

As is typical in optimization, a loss function is defined as follows.

Definition 1.1.4. *For a functional neural network $\mathcal{F}$ and a dataset $\{(\gamma_n(j), \delta_n(j))\}$ we say that the error for a given n is defined by*

$$E = \frac{1}{2} \int_{R^{(L)}} (\mathcal{F}(\gamma_n) - \delta_n)^2 \; dj_L \tag{1.1.9}$$

This error definition follows from $\mathcal{N}$ as the typical error function for $\mathcal{N}$ is just the square norm of the difference of the desired and predicted output vectors. In this case we use the L^2 norm on $C(R^{(L)})$ in the same fashion.

We first propose the following lemma as to aid in our derivation of a computationally suitable error backpropagation algorithm.

Lemma 1.1.5. *Given some layer, $l > 0$, in $\mathcal{F}$, functions of the form $\Psi^{(l)} = g'\left(\Sigma_l \sigma^{(l)}\right)$ are numerically integrable.*

Proof. If

$$\Psi^{(l)} = g'\left(\int_{R^{(l-1)}} \sigma^{(l-1)} w^{(l-1)} \; dj_{l-1}\right) \tag{1.1.10}$$

then

$$\Psi^{(l)} = g'\left(\sum_{b}^{Z_Y^{(l-1)}} j_l^b \sum_{a}^{Z_X^{(l-1)}} k_{a,b}^{(l-1)} \int_{R^{(l-1)}} \sigma^{(l-1)} j_{l-1}^a \; dj_{l-2}\right) \tag{1.1.11}$$

hence Ψ can be numerically integrated and thereby evaluated. □

The ability to simplify the derivative of the output of each layer greatly reduces the computational time of the error backpropagation. It becomes a function defined on the interval of integration of the next iterated integral.

Theorem 1.1.6. *The gradient, $\nabla E(\gamma, \delta)$, for the error function* (1.1.9) *on some $\mathcal{F}$ can be evaluated numerically.*

Proof. Recall that E over $\mathcal{F}$ is composed of $k_{x,y}^{(l)}$ for $x \in Z_X^{(l)}, y \in Z_Y^{(l)}$, and $0 \leq l \leq L$. If we show that $\frac{\partial E}{\partial k_{x,y}^{(l)}}$ can be numerically evaluated for arbitrary, l, x, y, then every component of ∇E is numerically evaluable and hence ∇E can be numerically evaluated. Given some arbitrary l in $\mathcal{F}$, let $n = L - l$. We will examine the particular partial derivative for the case that $n = 1$, and then for arbitrary n, induct over each iterated integral.

Consider the following expansion for $n = 1$,

$$\begin{aligned}
\frac{\partial E}{\partial k_{x,y}^{(L-n)}} &= \frac{\partial}{\partial k_{x,y}^{(L-1)}} \frac{1}{2} \int_{R^{(l)}} [\mathcal{F}(\gamma) - \delta]^2 \; dj_L \\
&= \int_{R^{(l)}} [\mathcal{F}(\gamma) - \delta] \, \Psi^{(L)} \int_{R^{(l-1)}} j_{L-1}^x j_L^y \sigma^{(L-1)} dj_{L-1} \; dj_L \\
&= \int_{R^{(l)}} [\mathcal{F}(\gamma) - \delta] \, \Psi^{(L)} j_L^y \int_{R^{(l-1)}} j_{L-1}^x \sigma^{(L-1)} dj_{L-1} \; dj_L
\end{aligned} \tag{1.1.12}$$

Since the second integral in (1.1.12) is exactly $I_x^{(L-1)}$ from (1.1.6), it follows that

$$\frac{\partial E}{\partial k_{x,y}^{(n)}} = I_x^{(L-1)} \int_{R^{(l)}} \left[\mathcal{F}(\gamma) - \delta\right] \Psi^{(L)} j_L^y \, dj_L \tag{1.1.13}$$

and clearly for the case of $n = 1$, the theorem holds.

Now we will show that this is all the case for larger n. It will become clear why we have chosen to include $n = 1$ in the proof upon expansion of the pratial derivative in these higher order cases.

Let us expand the gradient for $n \in \{2, \ldots, L\}$.

$$\frac{\partial E}{\partial k_{x,y}^{L-n}} = \int_{R^{(L)}} [\mathcal{F}(\gamma) - \delta] \Psi^{(L)} \underbrace{\int_{R^{(L-1)}} w^{(L-1)} \Psi^{(L-1)} \int \cdots \int_{R^{(L-n+1)}} w^{(L-n+1)} \Psi^{(L-n+1)}}_{n-1 \text{ iterated integrals}}$$
$$\int_{R^{(L-n)}} \sigma^{(L-n)} j_{L-n}^a j_{L-n+1}^b \, dj_{L-n} \ldots dj_L \tag{1.1.14}$$

As aforementioned, proving the $n = 1$ case is required because for $n = 1$, (1.1.14) has a section of $n - 1 = 0$ iterated integrals which cannot be possible for the proceeding logic.

We now use the order invariance properly of iterated integrals (that is, $\int_A \int_B f(x, y) \, dxdy = \int_B \int_A f(x, y) \, dydx$) and reverse the order of integration of (1.1.14).

In order to reverse the order of integration we must ensure each iterated integral has an integrand which contains variables which are guaranteed integration over some region. To examine this, we propose the following recurrence relation for the gradient.

Let $\{B_s\}$ be defined along $L - n \leq s \leq L$, as follows

$$\begin{aligned} B_L &= \int_{R^{(L)}} \left[\mathcal{F}(\gamma) - \delta\right] \Psi^{(L)} B_{L-1} \, dj_L, \\ B_s &= \int_{R^{(l)}} \Psi^{(l)} \sum_a^{Z_X^{(l)}} \sum_b^{Z_Y^{(l)}} j_l^a j_{l+1}^b B_{l-1} \, dj_l, \\ B_{L-n} &= \int_{R^{(L-n)}} j_{L-n}^x j_{L-n+1}^y \, dj_{L-n} \end{aligned} \tag{1.1.15}$$

such that $\frac{\partial E}{\partial k_{x,y}^{(l)}} = B_L$. If we wish to reverse the order of integration, we must find a reoccurrence relation on a sequence, $\{\mathfrak{B}_s\}$ such that $\frac{\partial E}{\partial k_{x,y}^{(L-n)}} = \mathfrak{B}_{L-n} = B_L$. Consider the gradual reversal of (1.1.14).

Clearly,

$$\begin{aligned} \frac{\partial E}{\partial k_{x,y}^{(l)}} = \int_{R^{(L-n)}} \sigma^{(L-n)} j_{L-n}^x \int_{R^{(L)}} [\mathcal{F}(\gamma) - \delta] \Psi^L \int_{R^{(L-1)}} w^{(L-1)} \Psi^{(L-1)} \\ \int \cdots \int_{R^{(L-n+1)}} j_{L-n+1}^y w^{(L-n+1)} \Psi^{(L-n+1)} \, dj_{L-n+1} \ldots dj_L dj_{L-n} \end{aligned} \tag{1.1.16}$$

is the first order reversal of (1.1.14). We now show the second order case with first weight

function expanded.

$$\frac{\partial E}{\partial k_{x,y}^{(l)}} = \int_{R^{(L-n)}} \sigma^{(L-n)} j_{L-n}^{x} \int_{R^{(L-n+1)}} \sum_{b}^{Z_Y} \sum_{a}^{Z_X} k_{a,b} j_{L-n+1}^{a+y} \Psi^{(L-n+1)} \int_{R^{(L)}} [\mathcal{F}(\gamma) - \delta] \Psi^{L} \int \cdots \int_{R^{(L-n+1)}} j_{L-n+2}^{b} w^{(L-n+2)} \Psi^{(L-n+2)} \, dj_{L-n+1} \ldots dj_L dj_{L-n}. \tag{1.1.17}$$

Repeated iteration of the method seen in (1.1.16) and (1.1.17), where the inner most integral is moved to the outside of the $(L-s)^{\text{th}}$ iterated integral, with s is the iteration, yields the following full reversal of (1.1.14). For notational simplicity recall that $l = L - n$, then

$$\frac{\partial E}{\partial k_{x,y}^{(l)}} = \int_{R^{(l)}} \sigma^{(l)} j_{l}^{x} \int_{R^{(l+1)}} \sum_{a}^{Z_X^{(l+1)}} j_{l+1}^{a+y} \Psi^{(l+1)} \int_{R^{(l+2)}} \sum_{b}^{Z_Y^{(l+1)}} \sum_{c}^{Z_X^{(l+2)}} k_{a,b}^{(l+1)} j_{l+2}^{b+c} \Psi^{(l+2)} \int_{R^{(l+3)}} \sum_{d}^{Z_Y^{(l+2)}} \sum_{e}^{Z_X^{(l+3)}} k_{c,d}^{(l+2)} j_{l+3}^{d+e} \Psi^{(l+3)} \int \cdots \int_{R^{(L)}} \sum_{q}^{Z_Y^{(L-1)}} k_{p,q}^{(L-1)} j_{L}^{q} \, [\mathcal{F}(\gamma) - \delta] \Psi^{(L)} \, dj_L \ldots dj_{L-n}. \tag{1.1.18}$$

Observing the reversal in (1.1.18), we yield the following recurrence relation for $\{\mathfrak{A}_s\}$. Bare in mind, $l = L - n$, x and y still correspond with $\frac{\partial E}{\partial k_{x,y}^{(l)}}$, and the following relation uses its definition on s for cases not otherwise defined.

$$\begin{aligned}
\mathfrak{A}_{L,t} &= \int_{R^{(L)}} \sum_{b}^{Z_Y^{(L-1)}} k_{t,b}^{(L-1)} j_{L}^{b} \left[\mathcal{F}(\gamma) - \delta\right] \Psi^{(L)} \, dj_L. \\
\mathfrak{A}_{s,t} &= \int_{R^{(s)}} \sum_{b}^{Z_Y^{(s-1)}} \sum_{a}^{Z_X^{(s)}} k_{t,b}^{(s-1)} j_{s}^{a+b} \Psi^{(s)} \mathfrak{A}_{s+1,a} \, dj_s. \\
\mathfrak{A}_{l+1} &= \int_{R^{(l+1)}} \sum_{a}^{Z_X^{(l+1)}} j_{l+1}^{a+y} \Psi^{(l+1)} \mathfrak{A}_{l+2,a} \, dj_{l+1}. \\
\frac{\partial E}{\partial k_{x,y}^{(l)}} = \mathfrak{A}_l &= \int_{R^{(l)}} j_{l}^{x} \sigma^{(l)} \mathfrak{A}_{l+1} \, dj_l.
\end{aligned} \tag{1.1.19}$$

Note that $\mathfrak{A}_{L-n} = B_L$ by this logic.

With (1.1.19), we need only show that $\mathfrak{A}_{L-n}$ is integrable. Hence we induct on $L - n \leq s \leq L$ over $\{\mathfrak{A}_s\}$ under the proposition that $\mathfrak{A}_s$ is not only numerically integrable but also constant.

Consider the base case $s = L$. For every t, because every function in the integrand of $\mathfrak{A}_L$ in (1.1.19) is composed of j_L, functions of the form $\mathfrak{A}_L$ must be numerically integrable and clearly, $\mathfrak{A}_L \in \mathbb{R}$.

Now suppose that $\mathfrak{A}_{s+1,t}$ is numerically integrable and constant. Then, trivially, $\mathfrak{A}_{s,u}$ is also numerically integrable by the contents of the integrand in (1.1.19) and $\mathfrak{A}_{s,u} \in \mathbb{R}$. Hence, the proposition that $s+1$ implies s holds for $l + 1 < s < L$.

Lastly we must show that both $\mathfrak{A}_{l+1}$ and $\mathfrak{A}_l$ are numerically integrable. By induction $\mathfrak{A}_{l+2}$ must be numerically integrable. Hence by the contents of its integrand $\mathfrak{A}_{l+1}$ must

also be numerically integrable and real. As a result, $\mathfrak{A}_l = \frac{\partial E}{\partial k_{x,y}^{(l)}}$ is real and numerically integrable.

Since we have shown that $\frac{\partial E}{\partial k_{x,y}^{(l)}}$ is numerically integrable, ∇E must therefore be numerically evaluable as aforementioned. This completes the proof. □

1.1.4 Continuous Error Backpropagation Algorithm

The logic of the proceeding proof required the establishment of $\{\mathfrak{A}_s\}$ as a sequence with a recurrence relation defining each component of the gradient on some coefficient in the network. Interestingly enough for L large enough certain elements of $\{\mathfrak{A}_s\}$ can be reused for different n. In order to more accurately describe this process, we propose the following algorithm for calculating gradients in stochastic gradient descent.

1. For all $\lambda \in \{1, \ldots, L\}$, calculate $\Psi^{(\lambda)}$.
2. Calculate $K - \nabla E \to K$ where K is the vector of all coefficient matrices by calculating each partial matrix over $l \in \{0, \ldots, L-1\}$
   (a) If $l = L - 1 \Leftrightarrow (n = 1)$, then store into the coefficient matrix
   $$k_{x,y}^{(L-1)} - I_x^{(L-1)} \int_{R^{(L)}} \left[\mathcal{F}(\gamma) - \delta\right] \Psi^{(L)} j_L^y \, dj_L \to k_{x,y}^{(L-1)} \tag{1.1.20}$$
   for every x, y.
   (b) If $l < L - 1 \Leftrightarrow n > 1$, then
      i. Ensure that $\mathfrak{A}_{l+2,t}$ from (1.1.19) is computed for all $t \in Z_X^{(l+1)}$
      ii. Then for all $x \in Z_X^{(l)}$ and $y \in Z_Y^{(l)}$,
         A. Compute
         $$\mathfrak{A}_{l+1} = \int_{R^{(l+1)}} \sum_a^{Z_X^{(l+1)}} j_{l+1}^{a+y} \Psi^{(l+1)} \mathfrak{A}_{l+2,a} \, dj_{l+1}. \tag{1.1.21}$$
         B. Compute
         $$\frac{\partial E}{\partial k_{x,y}^{(l)}} = \mathfrak{A}_l = \int_{R^{(l)}} j_l^x \sigma^{(l)} \mathfrak{A}_{l+1} \, dj_l. \tag{1.1.22}$$
         C. Update the weights such that
         $$k_{x,y}^{(l)} - \mathfrak{A}_l \to k_{x,y}^{(l)} \tag{1.1.23}$$

With the completion of the implementation, the theoretical definition of functional neural networks is complete.