A GENERAL APPROACH FOR LEARNING CONSTITUTIVE RELATIONSHIPS (PHYSICAL LAWS) FROM NEURAL NETWORKS

ANDREW TEMPLE, AARON STEBNER, PETER COLLINS, AND BRANDEN KAPPES

1. Introduction

In the main, a disconnect exists between physically-based equations that describe observations of complex phenomena and models derived from machine learning methods, including neural networks. This disconnect results in misunderstandings (at a minimum) and philosophical gulfs (worse) between the data scientists and those scientists deeply rooted in disciplines which traditionally study the various complex phenomena. On the one hand, many academics and scientists have called machine learning, data analytics, and neural networks "black boxes", implying that they are either not understood or that they are simply another, perhaps crude, tool. On the other hand, there are those

2. Methods

Fully dense neural network (NN) architectures, such as the one shown in Figure **??**, perform a sequence of affine transformations, $\mathbf{z}_i \leftarrow \boldsymbol{\theta}_i \mathbf{x}^{(i)}$, followed by element-wise functional operations, $\sigma(\mathbf{z}_i)$ to introduce non-linearity at each layer; that is, each layer stretches and distorts the underlying space.

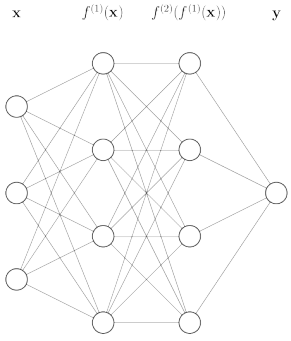

Figure 1. Schematic view of a fully dense neural network. Each sequence of affine and non-linear transformations are captured in the function, $f_i(\mathbf{x})$: $\mathbf{x}^{(i+1)} \leftarrow \sigma(\boldsymbol{\theta}_i \mathbf{x}^{(i)})$

The resulting network,

$$f(x) = \sigma(\boldsymbol{\theta}_n \sigma(\boldsymbol{\theta}_{n-1} \sigma(\dots \boldsymbol{\theta}_2 \sigma(\boldsymbol{\theta}_1 \mathbf{x})))) \tag{1}$$

is an arbitrary function generator, but at present, the network weights $\boldsymbol{\theta}_i$ can not map back to analytic forms that capture and describe the underlying physics. There are, however, many such mappings through polynomial series expansions,

$$f(x) = \sum_{n=0}^{\infty} a_n x^n \tag{2}$$

The variable $\mathbf{x}$ in Equation (**??**) represents a column vector containing $\boldsymbol{n}$ input elements, and thus, the output from each node is multplied by the respective weight before the application of the subsequent activation function, e.g.

$$\mathbf{z} = \theta^{(1)} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \Rightarrow \begin{bmatrix} \theta_{11}x_1 + \theta_{12}x_2 + \dots \theta_{1n}x_n \\ \theta_{21}x_1 + \theta_{22}x_2 + \dots \theta_{2n}x_n \\ \vdots \\ \theta_{m1}x_1 + \theta_{m2}x_2 + \dots \theta_{mn}x_n \end{bmatrix} \tag{3}$$

and $\boldsymbol{\sigma}(\mathbf{z})$ serves as the input of the next layer in the network, where $\mathbf{z} = (\boldsymbol{\theta}^{(1)}\mathbf{x})$.

We hypothesize that the physics of a process can be extracted by fitting the polynomial expansions of known physical relationships to the polynomial coefficients of a polynomial series expansion of Equation (**??**).

For example, consider the Softmax activation function which is given in Equation (???)

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}} \tag{4}$$

The goal is to obtain a activation generating function for the chosen activation function in the neural network.

The partion function, $\mathbf{Z}$, is defined below as

$$\mathbf{Z} = \sum_{j=1}^{k} e^{z_j} \tag{5}$$

The series representation for the exponential is given by the following

$$e^{z_i} = \sum_{k=0}^{\infty} \frac{1}{k!} z_i^k \tag{6}$$

Therefore

$$\sigma(\mathbf{z})_i = \sum_{k=0}^{\infty} \alpha_k z_i^k \tag{7}$$

Where $\alpha_k = \frac{1}{\mathbf{Z}k!}$ is the generating function for the Softmax

The ReLU (rectified linear units) function is another commonly used activation function and is given below in Equation (**??**):

$$f(x) = \begin{cases} x, & \text{if } x > 0 \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

The generating function for the ReLU is given in Equation (**??**) and it is dependent on both the input variables and the network weights.

$$\alpha_n = \begin{cases} 1 & \mathbf{z} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

Similarly, the linear generating function is given in Equation (**??**) but it is only dependent upon **n**.

$$\alpha_n = \begin{cases} 1 & \mathbf{n} = 1 \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

Although ReLU (rectified linear units) have become a more common activation function, its discontinuity at $x = 0$ requires an infinite series to fully capture the behavior at this transition.

However, the softplus function,

$$\begin{aligned}
f(x) &= log(1+e^x) \\
&= \frac{1}{\alpha} log(1+e^{\alpha x}), \quad \alpha > 0 \\
&= \frac{1}{\alpha} log(z), \quad z = 1+e^{\alpha x} \\
&= \frac{1}{\alpha} \sum_{k=1}^{\infty} \frac{1}{k} \left(\frac{z-1}{z} \right)^k \\
&= \frac{1}{\alpha} \sum_{k=1}^{\infty} \frac{1}{k} (z-1)^k z^{-k} \\
&= \frac{1}{\alpha} \sum_{k=1}^{\infty} \frac{1}{k} e^{\alpha x k} (1+e^{\alpha x})^k \\
&= \frac{1}{\alpha} \sum_{k=1}^{\infty} \frac{1}{k} \left(\sum_{m=0}^{\infty} \frac{\alpha^m x^m}{m!} \right)^k \left(1 + \sum_{n=0}^{k} \frac{\alpha^n x^n}{n!} \right)^{-k}
\end{aligned}$$

From Gradshteyn and Ryzhik - 0.314 Power series raised to powers.

$$\left(\sum_{k=0}^{\infty} a_k x^k \right)^n = \sum_{k=0}^{\infty} c_k x^k \tag{11}$$

where

$$c_0 = a_0^n, \quad c_m = \frac{1}{m a_0} \sum_{k=0}^{\infty} c_k x^k$$

And therefore it can be seen that

$$c_0 = \frac{\alpha^0}{0!} = 1, \quad c_l = \frac{1}{l} \sum_{j=1}^{l} (jk - l + j) \frac{\alpha^l}{l!} c_{l-j}$$

$$= \frac{1}{\alpha} \sum_{k=1}^{\infty} \frac{1}{k} \sum_{m=0}^{\infty} c_m x^m \left(1 + \sum_{n=0}^{\infty} \frac{\alpha^n x^n}{n!} \right)^{-k}$$

From Gradshteyn and Ryzhik - 1.1 Power of Binomials, 1.111 Power Series

$$(a+x)^n = \sum_{k=0}^{n} \binom{n}{k} x^k a^{n-k} \tag{12}$$

And similarly

$$= \frac{1}{\alpha} \sum_{k=1}^{\infty} \frac{1}{k} \sum_{m=0}^{\infty} c_m x^m \left[\sum_{i=0}^{k} \binom{k}{i} \left(\sum_{n=0}^{\infty} \frac{\alpha^n x^n}{n!} \right)^i \right]^{-1}$$

Once again the substitution for a power series raised to a power is applied.

$$d_0 = 1, \quad d_l = \frac{1}{l} \sum_{j=1}^{l} (ji - l + j) \frac{\alpha^l}{l!} d_{l-j}$$

$$= \frac{1}{\alpha} \sum_{k=1}^{\infty} \frac{1}{k} \left[\frac{\sum_{m=0}^{\infty} c_m x^m}{\sum_{i=0}^{k} \frac{1}{\binom{k}{i}} \sum_{n=0}^{\infty} d_n x^n} \right]$$

After some rearranging of term, the result is given below.

$$= \frac{1}{\alpha} \sum_{k=1}^{\infty} \frac{1}{k} \left[\frac{\sum_{m=0}^{\infty} c_m x^m}{\sum_{n=0}^{\infty} \sum_{i=0}^{k} \frac{1}{\binom{k}{i}} d_n x^n} \right]$$

And therefore the following is obtained.

$$= \frac{1}{\alpha} \sum_{k=1}^{\infty} \frac{1}{k} \left[\frac{\sum_{m=0}^{\infty} c_m x^m}{\sum_{n=0}^{\infty} d'_n x^n} \right]$$

Where

$$d'_n = \sum_{i=0}^{k} \frac{1}{\binom{k}{i}} d_n$$

From Gradshteyn and Ryzhik - 0.313 Division of power series.

$$\frac{\sum_{k=0}^{\infty} b_k x^k}{\sum_{k=0}^{\infty} a_k x^k} = \frac{1}{a_0} \sum_{k=0}^{\infty} c_k x^k \tag{13}$$

where

$$c_n + \frac{1}{a_0}\sum_{k=1}^{n} c_{n-k}a_k - b_n = 0 \tag{14}$$

And therefore

$$b_m = \sum_{j=1}^{m} c_m - b_{m-j}d_j$$

$$= \frac{1}{\alpha}\sum_{k=1}^{\infty}\frac{1}{k}\sum_{n=0}^{\infty} b_n x^n$$
$$= \sum_{n=0}^{\infty}\sum_{k=1}^{\infty}\frac{b_n}{\alpha k}x^n = \frac{1}{\alpha}log(1+e^{\alpha x})$$

where $\sum_{k=1}^{\infty}\frac{b_n}{\alpha k}$ is the generating function for the softplus

The element-wise exponent is used repeatedly in both the constitutive relation and neural network expansions. For the product of two matrices, $A : A \in \mathbb{R}^{m\times n}$ and $B : B \in \mathbb{R}^{n\times q}$, the element-wise exponent results in an expanded basis that explicitly includes the cross terms from $\mathbf{A}$ and $\mathbf{B}$.

$$
\begin{aligned}
(AB)^{\circ p} &= \left(\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & & \vdots \\ \vdots & & \ddots & \\ a_{m1} & \cdots & & a_{mn} \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1q} \\ b_{21} & b_{22} & & \vdots \\ \vdots & & \ddots & \\ b_{n1} & \cdots & & b_{nq} \end{pmatrix} \right)^{\circ p} \\
&= \begin{pmatrix} \mathbf{a}_{1k}\mathbf{b}_{k1} & \mathbf{a}_{1k}\mathbf{b}_{k2} & \cdots & \mathbf{a}_{1k}\mathbf{b}_{kq} \\ \mathbf{a}_{2k}\mathbf{b}_{k1} & \mathbf{a}_{2k}\mathbf{b}_{k2} & & \vdots \\ \vdots & & \ddots & \\ \mathbf{a}_{mk}\mathbf{b}_{k1} & \cdots & & \mathbf{a}_{mk}\mathbf{b}_{kq} \end{pmatrix}^{\circ p} \\
&= \begin{pmatrix} (\mathbf{a}_{1k}\mathbf{b}_{k1})^p & (\mathbf{a}_{1k}\mathbf{b}_{k2})^p & \cdots & (\mathbf{a}_{1k}\mathbf{b}_{kq})^p \\ (\mathbf{a}_{2k}\mathbf{b}_{k1})^p & (\mathbf{a}_{2k}\mathbf{b}_{k2})^p & & \vdots \\ \vdots & & \ddots & \\ (\mathbf{a}_{mk}\mathbf{b}_{k1})^p & \cdots & & (\mathbf{a}_{mk}\mathbf{b}_{kq})^p \end{pmatrix} \qquad \text{(15a)}
\end{aligned}
$$

$$
\begin{aligned}
(\mathbf{a}_{ik}\mathbf{b}_{kj})^p &= (a_{i1}b_{1j} + a_{i2}b_{2j} + \cdots + a_{in}b_{nj})^p \\
&= \sum_{k1+k2+\cdots+k_n=p} \binom{p}{k_1, k_2, \ldots, k_n} \prod_{m=1}^{n} (a_{im}b_{mj})^{k_m} \\
&= \sum_{\|\mathbf{k}\|_1=p} \binom{p}{k_1, k_2, \ldots, k_n} \prod_{m=1}^{n} (a_{im}b_{mj})^{k_m} \qquad \text{(15b)}
\end{aligned}
$$

(15c)

where summation over a repeated index is assumed. That is, Equation **??** is the expansion of the element-wise exponent of a vector product, $(\mathbf{ab})^{\circ p} = (\mathbf{ab})^p \neq \mathbf{a}^{\circ p}\mathbf{b}^{\circ p}$.

The analytical form, combining Equations (**??**) and (**??**), the estimated output of a two-layer NN can be written as an expansion:

$$
\begin{aligned}
\mathbf{y}_1 &= \sum_{k=0}^{\infty} a_k (\boldsymbol{\theta}_1^T \mathbf{x})^{\circ k} \\
\mathbf{y}_2 &= \sum_{k=0}^{\infty} b_k (\boldsymbol{\theta}_2^T \mathbf{y}_1)^{\circ k} \\
&= b_0 \mathbf{1} + \\
&\quad + b_1(\tilde{a}_0 + (\tilde{a}_1 + (\tilde{a}_2 + (\tilde{a}_3 + (\ldots)\tilde{\mathbf{x}})\tilde{\mathbf{x}})\tilde{\mathbf{x}})\tilde{\mathbf{x}}) \\
&\quad + b_2(\tilde{a}_0 + (\tilde{a}_1 + (\tilde{a}_2 + (\tilde{a}_3 + (\ldots)\tilde{\mathbf{x}})\tilde{\mathbf{x}})\tilde{\mathbf{x}})\tilde{\mathbf{x}})^2 \\
&\quad \vdots \\
&\quad + b_k(\tilde{a}_0 + (\tilde{a}_1 + (\tilde{a}_2 + (\tilde{a}_3 + (\ldots)\tilde{\mathbf{x}})\tilde{\mathbf{x}})\tilde{\mathbf{x}})\tilde{\mathbf{x}})^k \\
&\quad \vdots
\end{aligned} \tag{16}
$$

where $\tilde{a}_i = \boldsymbol{\theta}_2^T a_i$ and $\tilde{\mathbf{x}} = \boldsymbol{\theta}_1^T \mathbf{x}$. All $\boldsymbol{\theta}_i$, $\mathbf{x}$, and $\mathbf{y}$ are augmented to include the bias, $\mathbf{b}_i$, that is,

$$
\mathbf{x} \;:\; \mathbf{x} \leftarrow \begin{pmatrix} 1 \\ \mathbf{x} \end{pmatrix} = \begin{pmatrix} 1 \\ x_0 \\ x_1 \\ \vdots \\ x_n \end{pmatrix} \tag{17}
$$

$$
\mathbf{y} \;:\; \mathbf{y} \leftarrow \begin{pmatrix} 1 \\ \mathbf{y} \end{pmatrix} \tag{18}
$$

$$
\boldsymbol{\theta}_i \;:\; \boldsymbol{\theta}_i \leftarrow \begin{pmatrix} \mathbf{b}_i & \boldsymbol{\theta}_i \end{pmatrix} \tag{19}
$$

The element-wise exponent operator, $\mathbf{x}^{\circ m} = (x_1^m \; x_2^m \; \cdots \; x_n^m)^T$, raises each element of a vector or matrix to the specified power, m.

From Equation (**??**), $a_i = 0$ for $i = 2, 4, 6, \ldots$, and therefore,

$$
\begin{aligned}
\mathbf{y}_1 =& \sum_{k=0}^{\infty} a_k (\boldsymbol{\theta}_1^T \mathbf{x})^{\circ k} \\
\mathbf{y}_2 =& \sum_{k=0}^{\infty} b_k (\boldsymbol{\theta}_2^T \mathbf{y}_1)^{\circ k} \\
=& b_0 \mathbf{1} + \\
& + b_1(\tilde{a}_0 + (\tilde{a}_1 + (\tilde{a}_3 + (\tilde{a}_5 + (\ldots)\tilde{\mathbf{x}}^{\circ 2})\tilde{\mathbf{x}}^{\circ 2})\tilde{\mathbf{x}}^{\circ 2})\tilde{\mathbf{x}}) \\
& + b_2(\tilde{a}_0 + (\tilde{a}_1 + (\tilde{a}_3 + (\tilde{a}_5 + (\ldots)\tilde{\mathbf{x}}^{\circ 2})\tilde{\mathbf{x}}^{\circ 2})\tilde{\mathbf{x}}^{\circ 2})\tilde{\mathbf{x}})^{\circ 2} \\
& \vdots \\
& + b_k(\tilde{a}_0 + (\tilde{a}_1 + (\tilde{a}_3 + (\tilde{a}_5 + (\ldots)\tilde{\mathbf{x}}^{\circ 2})\tilde{\mathbf{x}}^{\circ 2})\tilde{\mathbf{x}}^{\circ 2})\tilde{\mathbf{x}})^{\circ k} \\
& \vdots \\
\mathbf{y}_2 =& \sum_{N=0}^{\infty} \sum_{k=0}^{N} \sum_{l=0}^{k} \sum_{m=0}^{l} \ldots b_N \binom{N}{k, l, m, \ldots} \tilde{a}_0^k \tilde{a}_1^l \tilde{a}_3^m \ldots \tilde{\mathbf{x}}^{\circ(N-k\ldots)} (\tilde{\mathbf{x}}^2)^{\circ(N-k-l\ldots)} (\tilde{\mathbf{x}}^2)^{\circ(N-k-l-m\ldots)} \\
& \qquad (20) \\
=& \sum_{N=0}^{\infty} \sum_{k=0}^{N} \sum_{l=0}^{k} \sum_{m=0}^{l} \ldots b_N \binom{N}{k, l, m, \ldots} \tilde{a}_0^k \tilde{a}_1^l \tilde{a}_3^m \ldots \tilde{\mathbf{x}}^{\circ(l+m+n+\ldots)} (\tilde{\mathbf{x}}^{\circ 2})^{\circ(m+n+\ldots)} (\tilde{\mathbf{x}}^{\circ 2})^{\circ(n+\ldots)}
\end{aligned}
$$

where $k + l + m + n + \ldots = N$. Collecting coefficients and terms of power k,

$$\mathbf{y}_2 = \sum_{k=0}^{\infty} c_k \tilde{\mathbf{x}}^{\circ k}$$

that, having the same form as Equation (**??**) creates a sequential process for determining the coefficients of the power series expansion of each layer in an ANN. Importantly, the output layer in a ANN regression is a single node with a linear activation, so the final layer, y_f, working from the last hidden layer, $\mathbf{y}_n$, is simply,

$$y_f = \boldsymbol{\theta}_n^T \mathbf{y}_n \qquad (21)$$

3. Discussion

Many non-trivial problems in materials science, and in science more broadly, are explained not through a single constitutive relationship, but through a combination of contributing physics.

The goal of this approach is to identify the coefficients of a hypothesized constitutive relationship, coefficients that capture the specific physics of a process, through a least-squares fit between the covector space of a neural network series expansion, $\boldsymbol{\alpha}(\boldsymbol{\theta})$, which is

a function of the model parameters, and the covector space of the constitutive relationship, $\boldsymbol{\beta}(C_k)$, which is a function of the physical constants of the model.

Having fit the model parameters, $\boldsymbol{\theta}$, on a vector space spanned by the column vectors of $\mathbf{x}$, the coefficients (the covector basis) of the neural network expansion, $\boldsymbol{\alpha}(\boldsymbol{\theta})$, capture the functional relationship between the input space and the response space, both affine and non-linear contributions, introduced through those parameters, $\boldsymbol{\theta}$, and the coefficient generating functions for the activation, e.g. Equations **??** (Rectified Linear Unit, ReLU) and **??** (softmax), respectively.

Naturally, the activation generating functions must match the activation function chosen in the neural network model architecture. Equation **??** is derived for ReLU activation, the most common hidden layer activation. (Generating functions for other activations are provided in the appendix.) In addition to the hidden layers, activation functions must also be chosen for the output layer. The two most common output activations are linear (Eq. **??**) and softmax (Eq. **??**) for regression and classification, respectively.

ReLU activation,

$$\sigma(z) = \begin{cases} z & \text{if } z > 0, \\ 0 & \text{otherwise} \end{cases}$$

is discontinuous in the first derivative at $z_i = 0$. Therefore, the coefficient generating function of this activation must either be either a function of the input data, $\mathbf{z}$ or a small modification must be made to the softplus,

$$\sigma(z;\alpha) = \frac{1}{\alpha}\ln\left(1 + e^{\alpha x}\right). \tag{22}$$

In the limit as α approaches infinity, this converges to the ReLU. Practically, though, α can be assigned a large value and the coefficient generating function no longer depends on the input data, see Equation **??** in the appendix. However, because of the high computational cost of expressing the coefficients using the modified softplus, and the relative low cost of forward evaluation of the trained neural network in order to apply Equation **??**, expressing the coefficient generating function of ReLU in terms of the input data, as in Equation **??**, rather than the training-data-agnostic approach in Equation **??** would seem more practical.

Condition 1. *Both the neural network and the constitutive relationship must depend on the same independent variables.*

That is, they must be described on the same basis vectors. The fit between the coefficients of the neural network expansion–$\boldsymbol{\alpha}$, the covector space of the neural network's basis vectors–and the coefficients of the series expansion of the constitutive relationship (its covector space, $\boldsymbol{\beta}$) is only possible because both span the same subspace and share a common description of the solution within that subspace, that is, the covector spaces are colinear. That is, suppose that A maps between a vector space and its covector space. If $A : A(X) = X^*$ and $A : A(Y) = Y^*$, then $X^* = Y^*$ if and only if $X = Y$.

If $\mathbf{x} = ([c], d)$ are the concentration of a solute and grain size, respectively, then the constitutive relationship expansion would be,

$$\begin{aligned}\sigma_y &= \sigma_f + C[c]^{2/3} + \frac{k}{\sqrt{d}} \\ &= \sigma_f + \underbrace{\sum_{k=0}^{\infty} a_k [c]^k}_{\text{solute}} + \underbrace{\sum_{k=0}^{\infty} b_k d^k}_{\text{Hall-Petch}} \\ &= \sigma_f + \sum_{k=0}^{\infty} a_k (\mathbf{S}_{\text{solute}}\mathbf{x})^{\circ k} + \sum_{k=0}^{\infty} b_k (\mathbf{S}_{\text{Hall-Petch}}\mathbf{x})^{\circ k} \end{aligned} \tag{23}$$

where

σ_f = Matrix flow stress.

a_k = Coefficient generating function for $x^{2/3}$.

b_k = Coefficient generating function for $x^{-1/2}$.

$\mathbf{S}_{\text{solute}}$ = Selection vector/matrix for choosing the solute concentration from the input vector.

$\mathbf{S}_{\text{Hall-Petch}}$ = Selection vector/matrix for choosing the grain size from the input vector.

(Coefficient generating functions can be found in Table **??**.) Combining Equations **??** and **??**,

$$\begin{aligned}\sigma_y = \sigma_f + \sum_{k=0}^{\infty} a_k \left[\sum_{l_1=0}^{k} \binom{k}{l_1, l_2} \prod_{m=1}^{2} ((\mathbf{S}_{\text{solute}})_m \mathbf{x}_m)^{l_m} \right] \\ + \sum_{k=0}^{\infty} b_k \left[\sum_{l_1=0}^{k} \binom{k}{l_1, l_2} \prod_{m=1}^{2} ((\mathbf{S}_{\text{Hall-Petch}})_m \mathbf{x}_m)^{l_m} \right], \; l_2 = k - l_1\end{aligned}$$

which further simplifies to

$$\begin{aligned}\sigma_y = \sigma_f + \sum_{k=0}^{\infty} \sum_{l_1=0}^{k} a_k \binom{k}{l_1, l_2} \prod_{m=1}^{2} (\mathbf{S}_{\text{solute}})_m^{l_m} \prod_{m=1}^{2} \mathbf{x}_m^{l_m} \\ + \sum_{k=0}^{\infty} \sum_{l_1=0}^{k} b_k \binom{k}{l_1, l_2} \prod_{m=1}^{2} (\mathbf{S}_{\text{Hall-Petch}})_m^{l_m} \prod_{m=1}^{2} \mathbf{x}_m^{l_m}\end{aligned}$$

such that the term $\prod_{m=1}^{2} \mathbf{x}_m^{l_m}$ serves as the common basis set over which the summation occurs, so that now, having a common basis, this simplifies to,

$$\sigma_y = \sigma_f + \sum_{k=0}^{\infty} \sum_{l_1=0}^{k} \binom{k}{l_1, l_2} \left(a_k \prod_{m=1}^{2} (\mathbf{S}_{\text{solute}})_m^{l_m} + b_k \prod_{m=1}^{2} (\mathbf{S}_{\text{Hall-Petch}})_m^{l_m} \right) \prod_{m=1}^{2} \mathbf{x}_m^{l_m} \tag{24}$$

It should be noted here that this common basis is a function of the length (dimension) of the input vector and of the order of the expansion. Therefore, two series will share the same basis vector and the same covector space if and only if they are taken to the same order. In addition, because there is no guarantee that the input vector space directions are orthogonal, there is no guarantee that the cross-term interactions will vanish and, therefore, must be included explicitly. This expansion includes all cross-terms and, through the element-wise exponentiation, also explicitly captures all combinations of powers of all cross-terms. Equation **??** shows that the polynomial series expansion for the first layer of a neural network,

$$y^{(1)} = \sum_k \alpha_k^{(1)} \left(\boldsymbol{\theta}^{(1)} \mathbf{x} \right)^{\circ k},$$

similarly relies on the element-wise exponential, $(\bullet)^{\circ n}$, as does the expansion of all layers. Unlike scalar exponentiation, element-wise exponentiation does not distribute, as seen in Equation **??**, and because element-wise exponentiation does not distribute, this equation explicitly captures all possible (second, $x_i^m x_j^n$; third, $x_i^m x_j^n x_k^p$; fourth, $x_i^m x_j^n x_k^p x_l^r$; etc.) cross-interactions of each term in $(\boldsymbol{\theta}\mathbf{x})$ at all polynomial orders. This is equivalent, then, to expanding over the basis set that includes all cross-interactions in the input vector space for both the constitutive relation and neural network polynomial expansions.

Data preprocessing is an important step in training a neural network to avoid implicit bias. Commonly, data is whitened, also known as scaling or standardization, $\mathbf{z}_s^{(k)} : \mathbf{z}_s^{(k)} = \frac{\mathbf{z}^{(k)} - \overline{\mathbf{z}^{(k)}}}{\sigma}$, where $\overline{\mathbf{z}^{(k)}}$ is the arithmetic mean and σ the standard deviation of data in layer, k. However, a model trained on such scaled data would no longer share the vector space of the constitutive relationship. To ensure that both the neural network and constitutive relation expansions share a common vector space, and thus a common covector space, this whitening procedure must be integrated into the construction of the neural network architecture.

Procedurally, the neural network expansion proceeds as described in Section **??**. The input to each layer, which is the output from the previous layer, is subjected first to an affine transformation, then to an activation function. The activation function is completely described though its polynomial expansion and the corresponding coefficient generating function. Using this same structure, then, data whitening can be applied to any layer, including the input layer, as an identity tranform ($\boldsymbol{\theta} = \mathbf{I}_d$, where d is the dimension of the source layer) followed by the whitening expansion whose coefficient generating function is simply,

$$\alpha_k(\mathbf{z}) = \begin{cases} -\overline{\mathbf{z}}/\sigma & \text{if } k = 0, \\ 1/\sigma & \text{if } k = 1, \\ 0 & \text{otherwise} \end{cases} \tag{25}$$

where

$\bar{\mathbf{z}}$ = The mean of the data into the standarization layer.

σ = The standard deviation of the data into the standardization layer.

By introducing such a whitening layer, data standardization can be included at any point in the neural network architecture.

TBW. This must answer the question: how do we know if we've measured the right things–not the number of measurements, but that we have enough information? How do we know that the solution has converged? Example: A model is to be fit to the number of cakes produced by a bakery. If we are given weights of flour and sugar and number of eggs, our model can accurately tell us the *volume* of cakes produced, but not the number. If this bakery makes cupcakes, but the model is trained across a spectrum of bakers, such as purveyors of wedding cakes and catering companies who work with large sheet cakes, then our dimensions (flour, sugar, eggs) are insufficient to fit the number of cakes produced. If, however, we also include number of orders and revenue, some information about the *quanta* of cakes is baked into those two additional dimensions (sorry, I couldn't help myself). Therefore, a model based only on (flour, sugar, eggs) is dimensionally insufficient, but a model based on (flour, sugar, eggs, order size, revenue) is dimensionally sufficient.

This is a broader question that may be beyond the scope of this paper. Let's return to this if, after completing the first pass, we feel that this can be addressed by what we've done.

This leads to a seven-step process for systematically and incrementally extracting physics information from an ANN:

(1) Collect data–features and targets–for which relationships are expected to exist.
(2) Design and train a fully dense multi-layer perceptron network (ANN).
(3) Build a power series expansion from the architecture of this ANN, using Equations (**??**) and (**??**) to populate the coefficients using the trained weights from the neural network.
(4) Hypothesize a constitutive relationship between the feature space and the target space.
(5) Recast the terms in the hypothesis function from #**??** as power series expansions, creating power series coefficient generating functions that are functions of the constitutive model fitting parameters. An example of this process is provided below, and a table of select power series expansions relevant to materials research are provided in Table (**??**).
(6) Perform an optimization, *e.g.* least squares, fit to find the fitting parameters from #**??**
(7) Calculate the residuals of the ANN power series expansion coefficient vector, and from this residual vector, the error in the model. If the accuracy is sufficient for the application, stop; otherwise, expand the constitutive relationship from step #**??** and repeat.

4. Conclusions

A generalized mathematical framework for proposing a constitutive relationship and fitting the physical constants associated with that constitutive relationship to a data corpus using the generalized fitting framework provided by machine learning/artificial intelligence has been presented. The proposed method maps between an arbitrary constitutive relationship and an artificial neural network model. The resulting covector spaces (coefficients) of the series are colinear. The generating function for the constitutive relationship is in terms of physical constants while that of neural network is determined through the trained model parameters. The method of least-squares is used to fit the physical constants to the trained neural network model parameters through this colinear covector space.

A simplified yield strength model, which includes flow stress, solute concentration, and Hall-Petch strengthening, is provided as an example to demonstrate the more general form for constructing the vector space for these two models.

Neural network series expansions are constructed layer-wise, which allows this framework to handle arbitrarily complex network architectures, including drop out, whitening, and any element-wise activation function. Mathematical descriptions of the rectified linear unit (ReLU) activation commonly used in neural network hidden layers, linear activation for regression networks, and softmax activation for classification networks are derived.

A mathematical framework to create constitutive relationship series expansions is also provided. Select coefficient generating functions for functional forms commonly found in materials physics are presented in the Appendix (Table **??**) to serve as a starting point, but any other functions that can be described by a polynomial series may be used.

As a single framework capable of describing arbitrarily complex relationships, this approach is intended to facilitate fits between existing data and any hypothesized constitutive relationships built upon the same vector space as the trained neural network model.

The authors wish to thank Linus, Martin, Richard, ... for fruitful discussions. Support through XYZ grant ABC123456 is gratefully acknowledged.

5. Appendix

5.1. **Activation Functions.**

5.1.1. *The logistic sigmoid.*

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{26}$$

is a special case of the generating function for the Euler polynomial coefficients,

$$\frac{2e^{xt}}{1+e^{t}} = \sum_{n=0}^{\infty} E_n(x)\frac{t^n}{n!} \tag{27}$$

where, for $x = 0$,

$$\sigma(x) = \frac{1}{2}\sum_{n=0}^{\infty} E_n(0)\frac{(-1)^n}{n!}. \tag{28}$$

The Euler polynomials at $x = 0$,

$$E_n(0) = -2(n+1)^{-1}\left(2^{n+1}-1\right)B_{n+1} \tag{29}$$

where B_n is the n^{th} Bernoulli number. Since Bernoulli numbers of odd index, with the exception of B_1, are zero, $E_i(0) = 0$ for $i = 2, 4, 6, \ldots, 2n$. Therefore, the summand and limits of Equation (**??**) change to

$$\sigma(x) = \frac{1}{2} - \frac{1}{2}\sum_{n=1}^{\infty}\left(\frac{E_{2n-1(0)}}{(2n-1)!}\right)x^{2n-1}. \tag{30}$$

The series representation of $E_{2n-1}(x)$

$$E_{2n-1}(x) = \frac{(-1)^n 4(2n-1)!}{\pi^{2n+1}}\sum_{k=0}^{\infty}\frac{\cos[(2k+1)\pi x]}{(2k+1)^{2n}} \tag{31}$$

such that,

$$E_{2n-1}(0) = \frac{(-1)^n 4(2n-1)!}{\pi^{2n+1}}\sum_{k=0}^{\infty}\frac{1}{(2k+1)^{2n}} \tag{32}$$

and therefore,

$$\begin{aligned}
\sigma(x) &= \frac{1}{2} - \sum_{n=1}^{\infty} 2\frac{(-1)^n}{\pi^{2n}}\left(\sum_{k=0}^{\infty}\frac{1}{(2k+1)^{2n}}\right)x^{2n-1} \qquad (33)\\
&= \frac{1}{2} - \sum_{n=1}^{\infty} 2\frac{(-1)^n}{\pi^{2n}}\left(4^{-n}\left(4^n-1\right)\zeta(2n)\right)x^{2n-1}\\
&= \frac{1}{2} - \sum_{n=1}^{\infty}\underbrace{2\left(\frac{-1}{4\pi^2}\right)^n\left(4^n-1\right)\zeta(2n)}_{a_n}x^{2n-1}\\
&= \sum_{n=0}^{\infty} a_n x^n,\ a_n = \begin{cases} 1/2 & n=0 \\ -2\left(\frac{-1}{4\pi^2}\right)^{(n+1)/2}\left(4^{(n+1)/2}-1\right)\zeta(n+1) & n \text{ odd} \\ 0 & n \text{ even} \end{cases} \qquad (34)
\end{aligned}$$

5.2. **Coefficient Generating Functions for Common Functions.** Central to this approach is the connection ability to represent a constitutive relationship and a neural network on the same vector/covector space. This is done through a polynomial series expansion of both the neural network, covered in the text, and the constitutive relationship. Select generating functions are provided here.

k	Ca^x	Cx^n	$Ce^{-\beta x}$	$Cx^{-1/2}$	$C(1+x)^\alpha$	$C\ln(1+x)$
0	1	–	C	C	C	0
1	$C\ln a$	–	$-\beta C$	$-\frac{1}{2}C$	$C\alpha$	C
2	$\frac{(\ln a)^2}{2}C$	–	$\frac{\beta^2}{2}C$	$\frac{3}{8}C$	$C\frac{\alpha(\alpha-1)}{2!}$	$\frac{-C}{2}$
$\vdots$			$\vdots$			
n	$\frac{(\ln a)^n}{n!}C$	$\begin{cases} C & \text{if } k=n \\ 0 & \text{otherwise} \end{cases}$	$(-1)^n\frac{\beta^n}{n!}C$	$C\prod_{i=1}^{n}(-1)\frac{2i-1}{2i}$	$C\frac{\prod_{i=1}^{n}\alpha-n+1}{n!}$	$C\frac{(-1)^{n+1}}{n}$
Constraint					$-1<x<1$	$-1<x\leq 1$