Advanced Modeling and Simulation in Engineering Sciences

RESEARCH ARTICLE Open Access

A methodology to assess and improve the physics consistency of an artificial neural network regression model for engineering applications

E. Rajasekhar Nicodemus*

*Correspondence: rajasekhar.nicodemus@gmail.com

Independent Researcher, 46-13-23, Devangula vari veedhi, Dondaparthy, Visakhaptnam 530016, Andhra Pradesh, India

Abstract

In recent times, artificial neural networks (ANNs) have become the popular choice of model for researchers while performing regression analysis between inputs and output. However; in scientific and engineering applications, developed ANN regression model is often found to be inconsistent with the physical laws. This is due to the fact that ANNs are purely based on data and do not have any understanding of underlying physical laws. Alternate ANN frameworks like PGNN (Physics guided neural network) has been proposed in literature which incorporate physics loss function in the overall loss function to partially alleviate this issue. However, these frameworks don't evaluate the physics consistency of relationship between inputs and output mapped by the ANN model which is the source of all physics inconsistencies. Hence, the present paper presents a methodology to assess and improve the physics consistency of the input output relationship mapped by the ANN regression model. The developed methodology can therefore be used to develop physics consistent ANN regression model. The heart of the methodology is an inferencing algorithm which interprets the input output relationship mapped by the ANN regression model. The inferencing algorithm is based on Taylor series and decomposes the ANN regression model into several region-wise polynomial models. Moreover, the inferencing algorithm can also find regions of singular zones in the ANN model predictions. The region-wise polynomial from inferencing algorithm can be used to assess the physics consistency of the ANN model. In the region of physics inconsistency, additional data points can be added and the ANN model can be re-trained. In cases, where the addition of data points is not possible, a physics based loss function can be used. The developed methodology is illustrated using several datasets. The developed methodology will help engineers and researchers built physics consistent ANN regression models.

Keywords: Neural network regression model, Taylor series, Radius of convergence, Degree of Taylor polynomial, Physics consistency, Physics loss function

Introduction

Artificial neural networks (ANNs) have been increasingly used in various fields such as engineering, geology, medicine, business, and experimental physics. Especially in science and engineering application, ANN regression models have been increasingly used to model complex relationship between various physics quantities. One of most common usage of ANN regression models in engineering applications are as surrogate models of physical processes. ANN surrogate models are generally developed either using experimental data or high fidelity computation-aided engineering (CAE) model data. Few of the applications of ANN regression models in engineering and science applications are detailed here. Wu et al. [1] developed an ANN surrogate model of engine air flow rate as a function of engine speed, manifold absolute pressure, intake and exhaust camshaft phasing and this model helped in accelerating the engine design process. Engine torque was maximized at elected engine speeds at a wide throttle opening by using ANN surrogate model of volumetric efficiency [2]. Meyer and Greff [3] established through investigations that ANNs have capability to replace conventional look-up tables of engine electronic control unit (ECU). Wendeker and Czarnigowski [4] proposed the use of an adaptive control system and a trained ANN surrogate model to minimize the error in the estimation of the required air–fuel ratio by an engine for stationary and dynamic operating conditions. An On-Board Diagnostic fault detection system using an ANN was developed by Grimaldi and Mariani [5]. Nicodemus and Sudipto [6] proposed a novel architecture of ANN regression model with end to end neuron connection to predict the finite element analysis (FEA) stress of an automobile connecting rod. An ANN regression model to predict to wear and tear of aircraft parts was developed by Paul et al. [7]. Recently, ANN regression models have also been used in string theory formulation [8, 9]. In all these applications [1–9], ANN regression models were used to model physical quantities which are continuous and differentiable and hence these studies used either tangent hyberbolic (tanh) or sigmoid activation function which results in smooth and differentiable ANN output function.

Furthermore, in these applications, it is essential that developed ANN model does not violate any known physics laws. However, since ANNs are unaware of the real-world physics which are being modelled and their predictions are only based on the available data it is shown in literature that ANNs can sometimes learn spurious relationships due to scarcity of data and noise. Liano [10] was among the first researchers to highlight that ANN are susceptible to learning from noise in data. Furthermore, Karpatne et al. [11] has shown that ANNs will learn relationships which are not consistent with physical laws but look deceptively good on both training and test when size of data is small.

To overcome these limitations, alternate frameworks of ANNs have been proposed in literature. Karpatne et al. [11] proposed a novel framework for combining scientific knowledge of physics-based models with ANNs and coined the term physics guided neural networks (PGNN) for the framework.. Two steps are generally involved in the framework of PGNN i.e. (a) constructing hybrid-physics-data models where output of the physical based model is also taken as an input (b) using physics-based loss functions which penalizes the ANN if physics laws are not maintained. Jia [12] proposed a similar physics guided NN framework for time series data called physics guided recurrent neural networks (PGRNN) for transient lake temperature modelling. Zhang et al. [13]

presented the physics-guided convolutional neural network (PhyCNN) for data-driven seismic response modelling. Wang et al. [14] proposed a PGNN methodology for modelling machine tool wear. Another class of neural network framework which are based on solving the physical laws rather than learning physics from training data are called physics informed neural networks (PINN). PINNs using the universal function approximator property of ANNs solve complex partial differential equation (PDE) systems [15, 16] without any data. The performance of PINNs is comparable to many established PDE solvers. However, PINN face similar disadvantages as physicals based models and require many calibrated parameters to capture the complex physics.

The PGNN frameworks are based on penalty loss for consistency of physical parameters outside the input variables and are not concerned about output relationship with input. However, the source for all physics inconsistencies is the improper relationship mapped by ANN between input variables and output variables. The complete physical relationship between inputs and output is generally unknown; however, partial knowledge from available literature and expert opinion is available for most of these relationships. For example, the physics relationship between air temperature and lake temperature in lake temperature modelling [11] is provided by Yu et al. [17]. But since ANNs map highly non-linear relationships in multidimensional input space it is very difficult to infer the mapped relationship of ANN between input and output variables and to assess whether the known partial knowledge is consistent with the developed ANN model or not.

There has been significant research and many developed methods to infer input output relationship of ANN classification model especially for computer vision problem whereas methods to infer input output relationship of ANN regression model are very scarce. Several authors proposed gradient based backpropgation algorithms [18–21] to infer feature importance of ANN classification model. It should be noted that the above algorithms [18–21] only work for ReLU activation function. However, most of ANN regression models [1–9] in engineering applications use tanh or sigmoid function hence these algorithm [18–21] cannot be used. Another method to visualize the pixel wise contribution to a class prediction is layer-wise relevance propagation (LRP) [22]. Sundarajan et al. [23] proposed a attribution method based on integrated gradients. Recently, Chattopadhyay et al. [24] considered ANN architecture as a structural causal model, and they presented a methodology to compute the causal effect of each feature on the output. These algorithms [22–24] provide a numerical value for contribution for each input feature for a particular output so these algorithms cannot be used to assess physics consistency of input output relation of ANN regression model. Another class of algorithm which interpret the input output relationship of ANN models are the methods based on perturbation of a particular output instance and building a local model around that instance. LIME (Local Interpretable Model-Agnostic Explanations) algorithm [25] is most notable among these algorithms. LIME algorithm is based on sampling the input space in neighbourhood of output point and building local interpretable model by minimizing weighted mean square loss function with weightage function based on distance metric. LIME is model agnostic i.e. no restricted on model being interpreted which could random forest, ANN, SVM (support vector machine) or any other machine learning(ML) model. Also LIME can provide explanations based on variables which need not be the input variables to ML model. LIME works best for classification where only relative importance of inputs are required for an output prediction. Whereas for ANN regression models the model characteristics is required to be done in certain region and LIME suffers from disadvantages of non-unique local model and local model dependence on parameters of weightage function.

The main aim of the work is to present a methodology to develop an ANN regression model which is consistent with the available physics knowledge. To do this an algorithm is required which would interpret the input output relationship mapped by the ANN regression model. Furthermore, ANN maps vastly different input and output relationships in different input regions. Hence, in the present work the input output relationship of ANN regression model is inferred by using a multiple region wise polynomial models for ANN regression model. One possible method for developing these polynomials is by fitting the polynomial function over the ANN function values; however, there would be error between ANN and polynomial function and polynomial function may not capture the relationship mapped by the ANN model. Therefore, Taylor series [26] will be the basis for the developed the region wise polynomial function. Taylor series since it is based on differentials of the function they can capture the primary relationship even with error between Taylor polynomial and ANN function. Taylor series share one key property with ANNs that is, both are universal approximators. In fact, both Taylor series and ANN have been used to same application of solving complex partial differential equations [15, 16, 27–29]. A key differentiator between Taylor series and ANN is that ANNs are flexible to approximate any function over any scale whereas Taylor series can approximate any function only inside the radius of convergence. A polynomial similar to Taylor polynomial can be generated using LIME algorithm but a key difference is that LIME builds the polynomial on top of ANN model by sampling input space in the vicinity of point of interest i.e. local model error behaviour would depend on model building process which include selection of sampling points and weightage function parameters. Whereas the Taylor polynomial in the current methodology doesn't another local model but characterizes the ANN model based on its local differentials at point of interest. Hence the local behaviour error profile will depend on characteristics of ANN model.

Methodology for developing a physics consistent ANN regression model

The traditional process of developing an ANN surrogate model in engineering applications involves the following steps as shown in Fig. 1. First the data points are sampled within the input space using either a latin hypercube or full factorial or other sampling techniques. Then the output data is generated using test procedure or CAE methodology for the selected sampled input points. The ANN regression model is fitted over the generated data. The predicted ANN output is correlated with generated data and regions in input space where accuracy is less, additional data are generated in that region. The process is iterated till the acceptable performance of model is reached. In this process, the accuracy of the predicted output is only considered.

In the proposed methodology, the ANN regression model will be optimized for prediction accuracy, physics consistency as well as consistency of singular zones in the input space as shown in Fig. 2. Mathematically, singular zones are defined as points where the function is not well-behaved. Physically, singularities are points in the

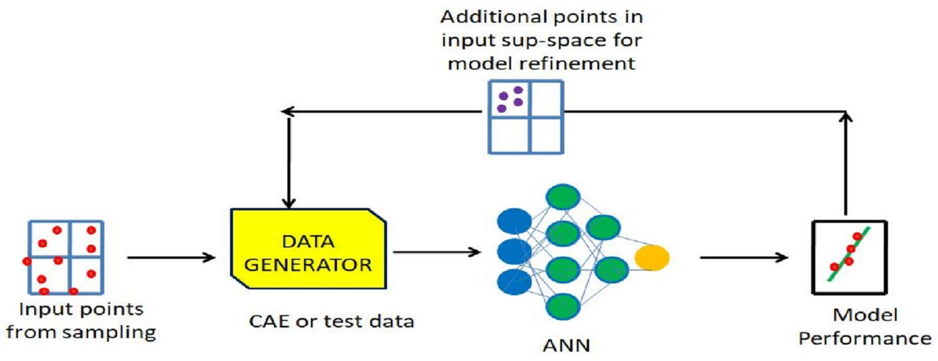

Additional points in input sup-space for model refinement
Input points from sampling
DATA GENERATOR
CAE or test data
ANN
Model Performance

Fig. 1 Process of developing an ANN surrogate model

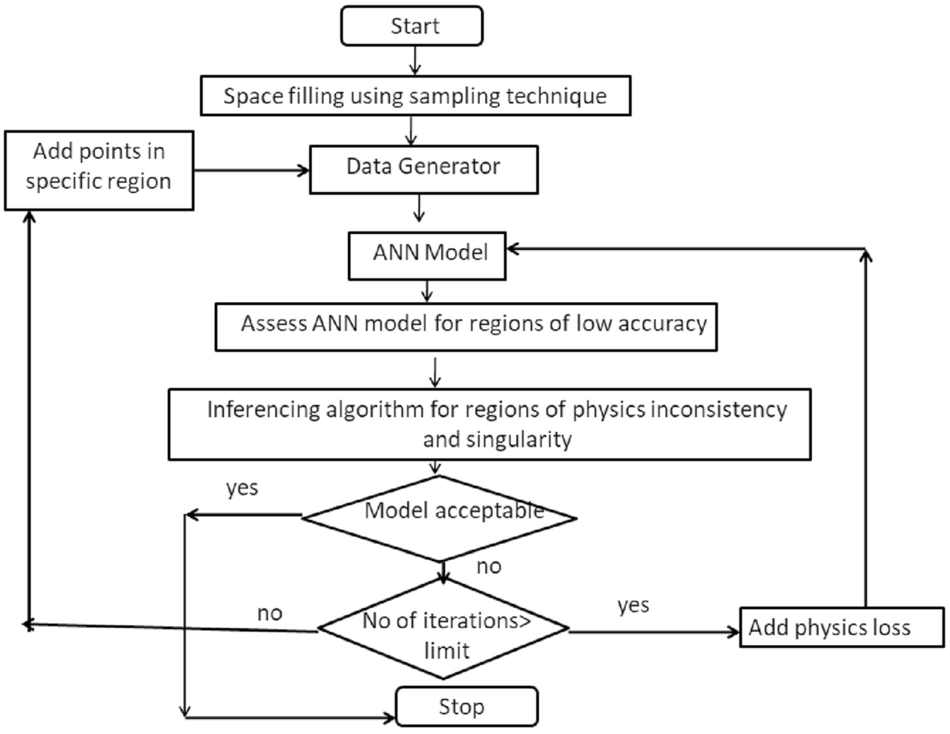

Start
Space filling using sampling technique
Add points in specific region
Data Generator
ANN Model
Assess ANN model for regions of low accuracy
Inferencing algorithm for regions of physics inconsistency and singularity
yes
Model acceptable
no
no
No of iterations> limit
yes
Add physics loss
Stop

Fig. 2 Flowchart for proposed methodology for developing physics consistent ANN models

input space where the output physical quantity behaves erratically and rapidly change in output is observed due to a very small change in one or more input variables. One of the well known physicals examples of singularity is resonance in vibrations. Generally, designers of engineering systems try to avoid running the physical system near singularity or design robust control system near singularity; hence identification of singularity region in the input space is of paramount importance to the user.

In next section, brief details regarding properties of Taylor polynomial of ANN function will be explained. Then the ANN inferencing algorithm is explained in two sub-sections. Finally applications of developed methodology on datasets will be presented.

Properties of Taylor polynomial of ANN function

Given a training dataset with inputs $\widehat{x} = \{x_1, x_2, x_{3,} \ldots\ldots\ldots\ldots x_k\}$ and target output $t(\widehat{x})$, let the ANN regression output function be $\mathrm{NN}(\widehat{x})$ and error function between ANN prediction and actual output be $\delta(\widehat{x})$ i.e.

$$\delta\left(\widehat{x}\right) = t\left(\widehat{x}\right) - \mathrm{NN}\left(\widehat{x}\right).$$

The properties of the output function of ANN ($\mathrm{NN}(\widehat{x})$) depends on the activation function used in the ANN. Using a finite higher order differentiable activation functions like tanh or sigmoid yields an ANN output function which is continuous and infinitely differentiable. Whereas using activation function like ReLU yield an ANN function which has zero higher order differentials. Since Taylor polynomial is based on higher order differentials, the present inferencing algorithm can be only used with activation functions like tanh and sigmoid. This is not a major hindrance since ReLU activation function is not used for ANN regression models in science and engineering applications. The output function predicted by ANN ($\mathrm{NN}(\widehat{x})$) using a finite higher order differentiable activation function can be approximated using Taylor polynomial of degree p at point $\widehat{x_o}$ and is given as

$$\begin{aligned}\mathrm{NN}\left(\widehat{x}\right) = &\sum_{n_1=0}^{p}\sum_{n_2=0}^{p}\sum_{n_3}^{p}\cdots\sum_{n_k=0}^{p}\frac{(x_1-x_{o1})^{n_1}(x_2-x_{o2})^{n_2}\ldots(x_k-x_{ok})^{n_k}}{n_1!n_2!\ldots\ldots\ldots n_k!}\\ &\frac{\partial^{n_1+n_2+n_3..+n_k}NN}{\partial {x_1}^{n_1}\partial {x_2}^{n_2}..\partial {x_k}^{n_k}}(x_{o1},x_{o2},..x_{ok})n_1+n_2+n_3..+n_k \le p.\end{aligned} \tag{1}$$

The Taylor polynomial is only accurate if the computation point ($\widehat{x}$) lies within the radius of convergence of development point ($\widehat{x_o}$). The various differential of $\mathrm{NN}(\widehat{x})$ with respect to inputs $\widehat{x}$ can be computed by using Tensorflow gradients function[15] which is an automatic differentiator. One thing to note here is that Taylor approximation is for the ANN function which includes the error function ($\delta\left(\widehat{x}\right)$) along with output function. It is well known fact that higher ANN function differentials don't correlate well with target function differentials even when using differentials in training (Sobolov training [30]). However, as explained above the ANN differentials are being used to characterize the ANN function and not target function and hence higher differentials trust worthiness is not an issue in the study. Several instances of use of higher order ANN differentials for reconstruction of ANN function has been shown throughout the manuscript even for datasets with noise. In subsequent sections properties of ANN differentials and Taylor polynomial are illustrated using examples. For the sake of simplicity examples used below are functions of single input but the same principles can be extended to multi inputs functions.

There are many aspects of the ANN model which should be consistent with the physics laws; however, the current paper only deals with physics consistency in relationship between input and output variables mapped by ANN regression model. Generally, in engineering applications the ANN regression model are used to compute the output variable variation in an interested region of input sub-space for decision making process. Hence the relationship between input and output variable are required for certain region for ANN regression model instead of a single point as in case ANN classification model. Therefore, the methodology to develop Taylor polynomial approximation for certain region of input sub-space is outlined in next sections. Each coefficient (sign and value) of Taylor polynomial explains relationship between input and output variable. These relationships can be checked for physics consistency based on available literature or expert opinion. With help of couple of examples the relationship explained by each Taylor coefficient will be illustrated here. Let t is target variable be function of two variables $\{x_1, x_2,\}$ i.e. t = NN (x_1, x_2) and let the p^th degree Taylor polynomial $C_{00} + C_{10}x_1 + C_{01}x_2 + C_{20}x^2{}_1 + C_{11}x_1x_2 + C_{02}x^2{}_2 \ldots\ldots\ldots\ldots\ldots\ldots + C_{0p}x^p{}_2$ which approximates target function in certain region of input space. Positive sign for Taylor coefficient C_{10} indicates target variable t increases with increase in x_1 and decreases with decrease in value of x_1 whereas negative sign indicates inversely proportional relationship between t and x_1 in the approximated region. The coefficient C_{11} explains the interaction effect of x_1 and x_2 on t i.e. if coefficient is positive then the target value increases with increase in x_1 and x_2 but simultaneous decrease in x_1 and x_2 will also yield increase in value of t. C_{20} shows the effect absolute value of x_1 on t. The values of coefficient demonstrate the relative weightage of these different effects. The explanations can be checked with known knowledge of system being modelled**.** For example, the ANN surrogate model of Ref. [1], models engine air flow rate and one of its input is engine speed. It is a well known fact that the engine air flow rate increases with engine speed irrespective of other input parameters values and the consistency of this physics relationship can be checked by the Taylor polynomial. An more detailed illustration of this physics inferencing is presented in Sect. 5.3.

Higher order differentials of ANNs

In order to understand the behaviour of higher order differentials of ANN function, let's take a simple ANN with one input and one hidden layer of "l" neurons and one output. Let's weights and biases for input layer to hidden layer be given as $w_{ih1}, w_{ih2}, \ldots\ldots\ldots w_{ihl}$ and $b_{ih1}, b_{ih2}, \ldots\ldots\ldots b_{ihl}$ and hidden layer to output layer be given as $w_{ho1}, w_{ho2}, \ldots\ldots\ldots w_{hol}$ and $b_{ho1}, b_{ho2}, \ldots\ldots\ldots b_{hol}$. Let the hidden layer have an activation function $f_1(.)$ and output layer is a linear layer. The output of each of l neurons is given as

$$h_1 = f_1(w_{ih1} * x + b_{ih1}), h_2 = f_1(w_{ih2} * x + b_{ih2}), h_l = f_1(w_{ihl} * x + b_{ihl})$$

The output function $\mathrm{NN}(\widehat{x})$ can be given as

$$\mathrm{NN}(\widehat{x}) = \sum_{i=1}^{l} w_{hoi} * h_i + b_{hoi}. \tag{2}$$

It can be seen from Eq. (2) that for infinitely differentiable activation functions $(f_1(.))$ like tanh or sigmoid function the ANN function $(\mathrm{NN}(\widehat{x}))$ will be infinitely differentiable and the function higher differentials will never vanish to zero. This is true even when data is generated from simple functions like x^2 and x^5 whose higher differentials vanish to zero. For activation function like ReLU whose higher order differentials from second differential is zero, the ANN function will have higher order differentials as zeroes. The same conclusions can be extended to ANN with multiple hidden units and in practice it holds true.

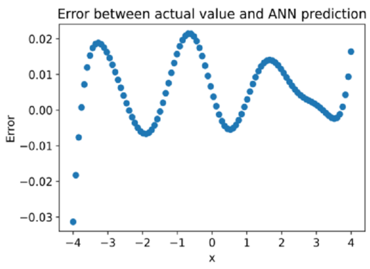
To visualize the behaviour of higher order differentials of ANN function, two artificial datasets were used to train ANN with ReLU and tanh activation function. 100 uniform data points for x were taken from interval [− 4, 4] and target output y is generated using two analytical functions Sin(x) and x^2. The ANN used here has 3 hidden layer with each layer having 15 neurons. The datasets can be modelled with lesser hidden layers and lesser number of neurons but these values were chosen so the observed behaviour would be valid for most of practically ANNs used for regression. For all datasets in the manuscript, the inputs and output values are normalized to have mean value of zero and variance value of one before feeding to the ANN model. Total data was split into 70–15–15 training, validation and test data sets and MSE along with regularization loss was taken as loss function and ANN was trained using gradient decent. It is well know that the first order differential is discontinuous and its higher order differentials are zero for output function of ANN while using ReLU activation function. This has been confirmed by the results from example ANN model using ReLU activation function but has not been shown in manuscript for sake of brevity. Hence, Taylor polynomial cannot be used to approximate ANN function when using ReLU activation function. Figure 3 shows the results from the ANN function while using tanh activation function which includes error function and some higher derivatives. It can be observed from Fig. 3 that the derivates from ANN function are continuous when using tanh activation and hence the ANN function can be approximated using Taylor polynomial.

Degree of Taylor polynomial

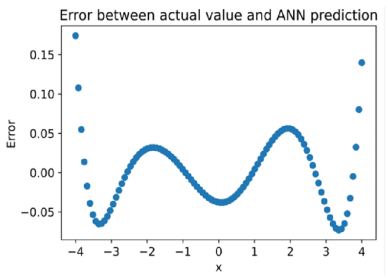
A zero degree Taylor polynomial is the function value at the development point itself and has zero radius of convergence. Increasing the degree of Taylor polynomial increases the radius of convergence at the development point. Radius of convergence is the distance from development point beyond which the Taylor series doesn't converge to the function value. However, computing radius of convergence is computationally expensive and hence a simple method would be used to compute the acceptable region of approximation by selecting a threshold error and comparing it with error between Taylor polynomial and ANN function. If the error is less than the threshold error than Taylor polynomial can be used to approximate in that region.

To understand the effect of degree of Taylor polynomial on approximation capabilities of Taylor polynomial, an artificial data set similar to previous studies [31] is generated with 200 uniform points in the range of x of [− 7.5,7.5] and target output y is generated using $\frac{Sin(x)}{x}$. Same 3 layers ANN architecture with tanh activation function is used. The error between Taylor polynomial and ANN function for different degrees of Taylor polynomial along with error function is plotted in Fig. 4. It can be seen from Fig. 4 that error decreases as degree Taylor polynomial increases. The acceptable region of approximation of particular degree of Taylor polynomial can be accessed by taking a suitable threshold error such as 0.02 or 0.05.

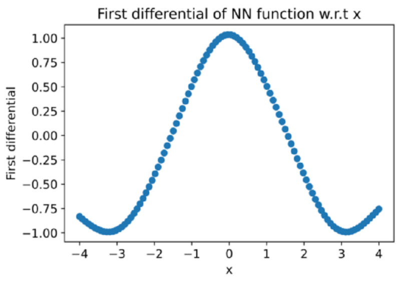
For a multi-variable input ANN function, the acceptable region of approximation may be different in different input variable directions. Also it is very difficult if not impossible



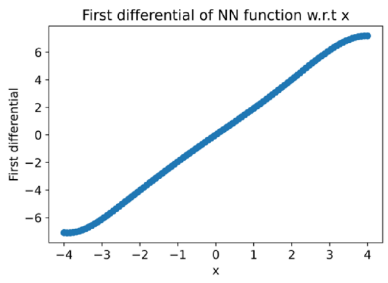
a. Error function ($\delta(\hat{x})$) for Sine data



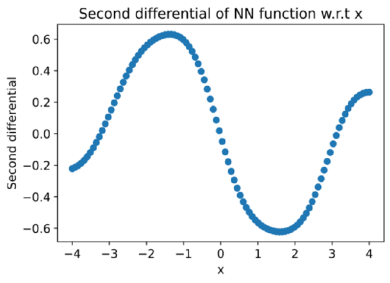
b. Error function ($\delta(\hat{x})$) for polynomial (x^2) data



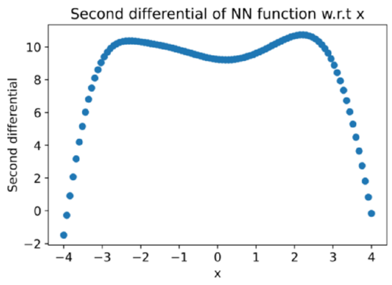
c. First order differential for Sine data



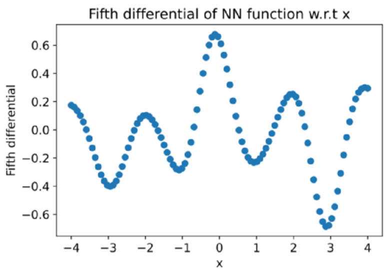
d. First order differential for polynomial (x^2) data



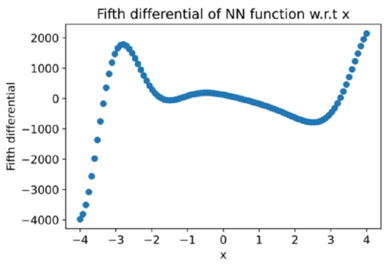
e. Second order differential for Sine data



f. Second order differential for polynomial (x^2) data



g. Fifth order differential for Sine data



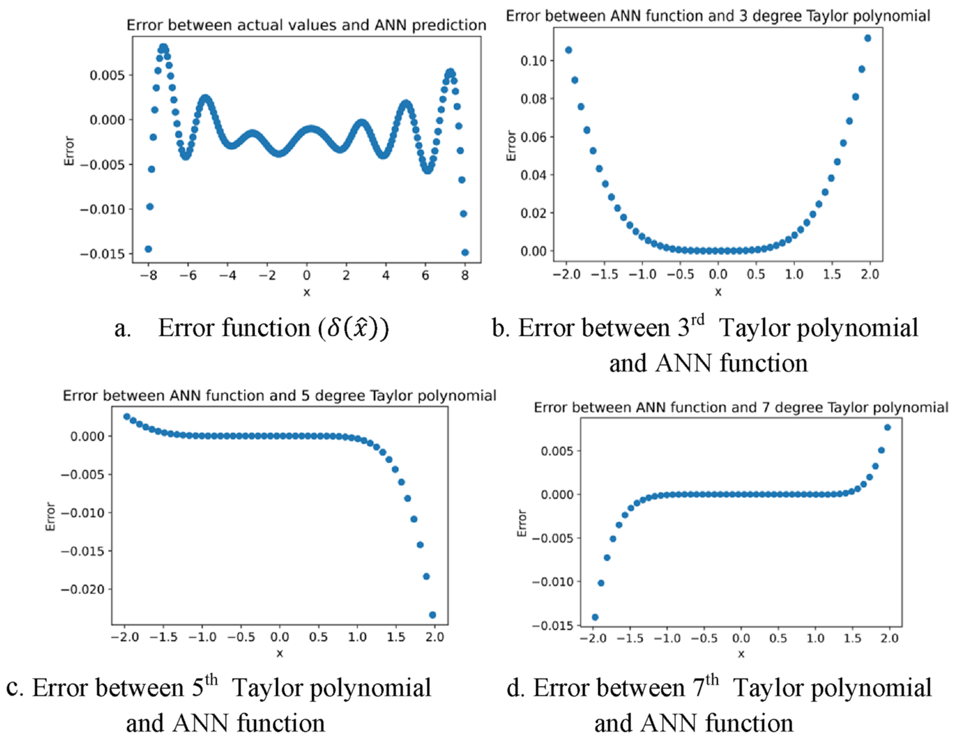
h. Fifth order differential for polynomial (x^2) data

Fig. 3 Results of ANN output function using tanh activation function

to visualize the error function or ANN function as function of input variables for a multi-variable.

Singularities in ANN functions

Taylor series cannot approximate function accurately in the singularity region. Radius of convergence of Taylor polynomial is severely reduced if development point is near the singularity. To show the Taylor polynomial behaviour near singularity, an artificial data with singularity is generated with 100 uniform points in the range of x of [− 4, 4]



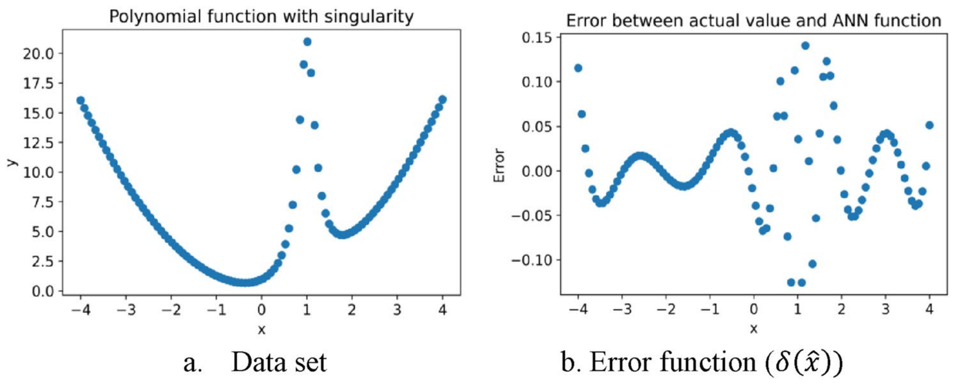
a. Error function ($\delta(\hat{x})$)

b. Error between 3rd Taylor polynomial and ANN function

c. Error between 5th Taylor polynomial and ANN function

d. Error between 7th Taylor polynomial and ANN function

Fig. 4 Errors of Taylor polynomials of different degrees



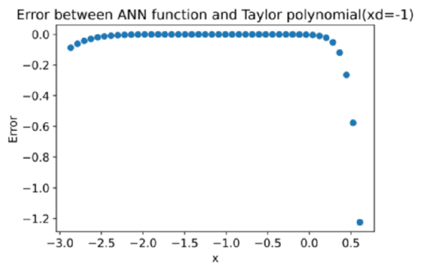
a. Data set

b. Error function ($\delta(\hat{x})$)

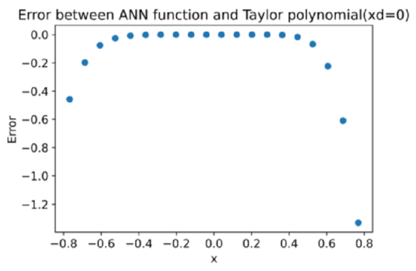
Fig. 5 Data set and error function of Polynomial function with singularity

and target output y is generated using $x^2 + \frac{1}{(x-1)^2+0.05}$. Same 3 layers ANN architecture with tanh activation function is used. The generated dataset and error function are shown in Fig. 5. It can be seen from Fig. 5a that the singularity zone arises around x=1 and also it difficult to predict the singularity just from observing error function plot in Fig. 5b.

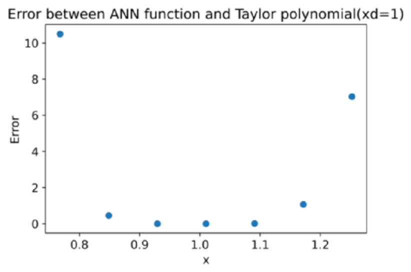
Taylor polynomial of 7th degree is used to approximate the ANN function with 3 different development points (x_d). The three development points are chosen such that first point is far away from singularly, second point is near to the singularity and third point is at the singularity. The error between ANN function and Taylor polynomial for



a. Error between ANN function and Taylor polynomial developed at $x_d = -1$



b. Error between ANN function and Taylor polynomial developed at $x_d = 0$



c. Error between ANN function and Taylor polynomial developed at $x_d = 1$

Fig. 6 Error between ANN function and Taylor polynomial using different development points

these 3 development points is shown in Fig. 6. For the development point which is far from singularity, the acceptable region of approximation is more in the direction opposite to singularity as compared to the direction towards the singularity. Moreover, it can be observed the acceptable region of approximation significantly reduces as distance between development points and singularity reduces. Taking the development point at singularity the acceptable region of approximation is very small and the error increases very sharply even at a very small distance from development point. Hence, using the Taylor polynomial the singularities in ANN function can be identified.

ANN regression model inferencing algorithm

In this section, the ANN inferencing algorithm will be presented. The inference algorithm will decompose ANN function into several region-wise Taylor. The algorithm is be divided into two major sections namely (a) computation of acceptable region of approximation of a Taylor polynomial for a given development point, and (b) strategy for selection of development points to envelope the input sub-space.

Computation of acceptable region of approximation for a ANN Taylor polynomial with a given development point

The computation of acceptable region of approximation is significantly complicated for multi variable function as compared to a single variable function. For even a single variable function as seen from Fig. 6 the acceptable region of approximation is different in different directions i.e. $x_d + \Delta x$ and $x_d - \Delta x$. Given a multi-variable ANN functions

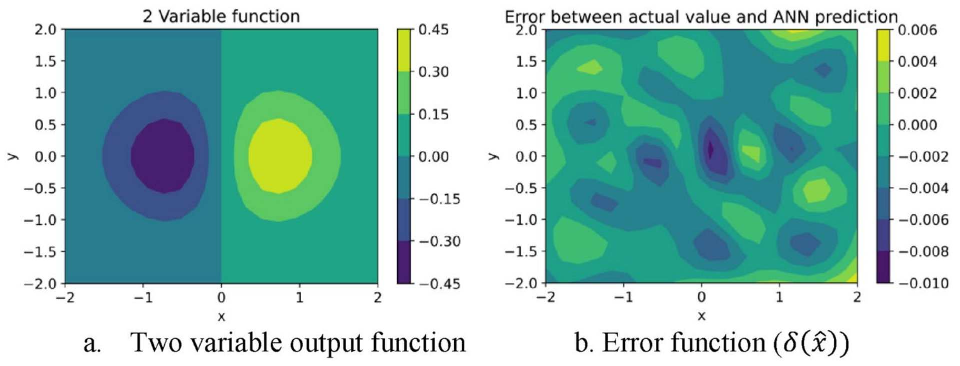


Fig. 7 Two variable output and error function

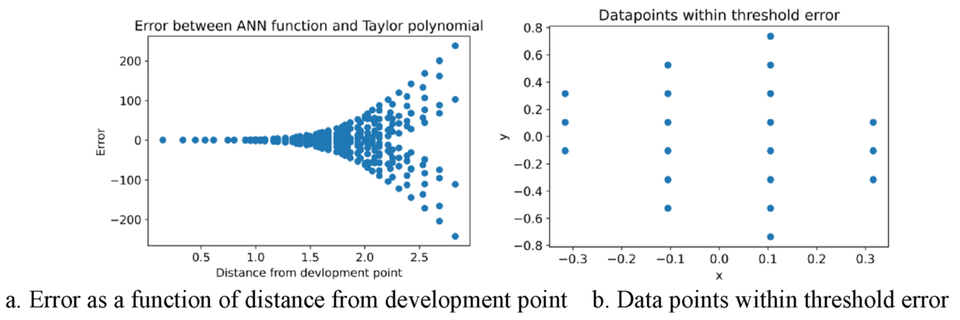


Fig. 8 Behaviour of error in 2 variable input space

$NN\{x_1, x_2, x_3, \ldots\ldots\ldots\ldots x_k\}$ with a development point $\{x_{1d}, x_{2d}, x_{3d}, \ldots\ldots\ldots\ldots x_{kd}\}$ the possible directions of traversals would be infinite and may be represented by any possible unit vector in the input space.

To illustrate the behaviour of error function between ANN function and Taylor polynomial around the development point for a multi-variable function a two variable data set is generated similar to Ref. [32]. A uniform grid of 400 points are generated within the range of x and y taken as [-2, 2] and output target points z is generated using the function $x * \exp(-(x^2 + y^2))$.Same ANN architecture with tanh activation function is used as before. The output function and error function for two variable dataset is shown in Fig. 7.

Let the threshold error for computing acceptable region of approximation be 0.005. In this study, absolute error is considered as threshold error, but the algorithm is also applicable when relative error or percentage error is considered as threshold error. The Taylor polynomial is computed at the input data points and error between ANN function and Taylor polynomial as a function of distance from development point is shown in Fig. 8a. It can be seen as the distance from development increases the error also increases. Furthermore, for the same distance from development point different values of errors are observed based on the direction of the evaluation point from development point. In Fig. 8b, the data points are shown for which the error is within the threshold error. An important thing to observe from Fig. 8b is that the acceptable region of approximation is not unique i.e. region A of $x \in [-0.1, 0.1]$ and $y \in [-0.6, 0.6]$ or region B of $x \in [-0.3, 0.3]$ and $y \in [-0.2, 0.2]$ can both be considered as acceptable region of approximation.

Based on observed behaviour an algorithm is developed and presented below to compute the acceptable region of approximation for a Taylor polynomial of a multi-variable ANN functions $NN\{x_1, x_2, x_{3,} \ldots\ldots\ldots\ldots x_k\}$ with a given development point $\{x_{1d}, x_{2d}, x_{3d,} \ldots\ldots\ldots\ldots x_{kd}\}$.

```
Algorithm:
Step 0: Check all inputs for consistency
Step 1: Repeat step 2-3 for i=1 to k (number of input variables)
Step 2: Compute the error of the Taylor polynomial in dimension $x_i$ around the development point i.e. compute error or Taylor polynomial at point's $\{x_1, x_2, x_{3,} \ldots, x_i + \Delta x_i \ldots\ldots\ldots x_k\}$ ($\Delta x_i$ should take positive and negative values)
Step 3: Compute maximum $\Delta x_i$ within which the error is less than threshold error in both positive and negative directions of $\Delta x_i$ and select minimum of those two values and it is the maximum or limiting traversal length in each input i direction ($xlim_i$)
Step 4: Develop a regular grid enveloping the domain defined by $x_1 \epsilon [x_{1d} - xlim_1, x_{1d} + xlim_1]$, $x_2 \epsilon [x_{2d} - xlim_2, x_{2d} + xlim_2]$, $x_3 \epsilon [x_{3d} - xlim_3, x_{3d} + xlim_3] \ldots x_k \epsilon [x_{kd} - xlim_k, x_{kd} + xlim_k]$. If Taylor polynomial error for the grid is within threshold error then stop the algorithm and this region is acceptable region of approximation else go to next step. For simplicity let us represent the traversal length in each direction as $\Delta x_{ai}$ and since this can be considered as zero iteration the limiting lengths can be denoted as $\Delta x_{ai0} = xlim_i$.
Step 5: Generate two acceptable and not-acceptable length arrays $\Delta xc_{ai}$ and $\Delta xnc_{ai}$ which represents the traversal lengths in each variable direction within threshold error and greater than threshold error. Initialize $\Delta xc_{ai}$ as zeroes and $\Delta xnc_{ai}$ as $xlim_i$.
Step 5: Repeat step 6 to 8 (iteration j) until convergence criterion is met
Step 6: Generate a grid of data points using ANN function for region $x_1 \epsilon [x_{1d} - \Delta x_{a1j}, x_{1d} + \Delta x_{a1j}]$, $x_2 \epsilon [x_{2d} - \Delta x_{a2j}, x_{2d} + \Delta x_{a2j}]$, $x_3 \epsilon [x_{3d} - \Delta x_{a3j}, x_{3d} + \Delta x_{a3j}] \ldots\ldots\ldots\ldots x_k \epsilon [x_{kd} - \Delta x_{akj}, x_{kd} + \Delta x_{akj}]$ and compute error of Taylor polynomial in the region where $\Delta x_{aij} = (\Delta xc_{ai} + \Delta xnc_{ai})/2$.
Step 7: If region is within threshold error
        $\Delta xc_{ai} = \Delta x_{aij}$
        else
        $\Delta xnc_{ai} = \Delta x_{aij}$
Step 8: Check convergence criteria. If a convergence criterion is met go to step 9 otherwise got to step 6
Step 9: Output the acceptable region of approximation as $x_1 \epsilon [x_{1d} - \Delta xc_{a1j}, x_{1d} + \Delta xc_{a1j}]$, $x_2 \epsilon [x_{2d} - \Delta xc_{a2j}, x_{2d} + \Delta xc_{a2j}]$, $x_3 \epsilon [x_{3d} - \Delta xc_{a3j}, x_{3d} + \Delta xc_{a3j}] \ldots\ldots x_k \epsilon [x_{kd} - \Delta xc_{akj}, x_{kd} + \Delta xc_{akj}]$
```

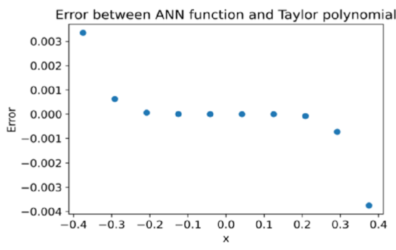
The convergence criteria can be a limit on maximum iterations or a minimum limit on distance between Δxc_{ai} and Δxnc_{ai}. If the limit on maximum iterations has reached and still algorithm couldn't find any non-zero region within the threshold error or detected region size is too small based on criteria then the development point can be considered as a singularity. The algorithm will be illustrated on two variable dataset at development point (0, 0). The algorithm is illustrated on the two variable data. Initially, 5th degree Taylor polynomial is computed in x and y direction by keeping other co-ordinate(s) constant at the development point. The error limits in x and y directions i.e. $xlim_1$ and $xlim_2$ are estimated as 0.375 and 0.8 for threshold error of 0.05. The error between ANN function and Taylor polynomial as a function of distance from development point in x and y direction is shown in Fig. 9a, b. A uniform grid of 100 points is generated in initial region of $x \in [-0.375, 0.375]$ and $y \in [-0.8, 0.8]$ and maximum absolute error for this region is computed. For this case the computed error is greater than threshold error and the first iteration region is computed as $x \in [-\frac{0.375+0}{2}, \frac{0.375+0}{2}]$ and $y \in \left[-\frac{0.8+0}{2}, \frac{0.8+0}{2}\right]$ i.e.$x \in [-0.1875, 0.1875]$ and $y \in [-0.4, 0.4]$. In this first iteration region, again a uniform grid of 100 points is generated and error of Taylor polynomial is computed. In the first iteration region, the error is less than the threshold error and hence the second iteration region is computed as $x \in [-\frac{0.375+0.1875}{2}, \frac{0.375+0.1875}{2}]$ and $y \in \left[-\frac{0.8+0.4}{2}, \frac{0.8+0.4}{2}\right]$ i.e.$x \in [-0.28125, 0.28125]$ and $y \in [-0.6, 0.6]$.The same process is continued till convergence criteria is met. The error profiles for region generated by the algorithm are shown for initial, first and fifth iterations are shown in Fig. 9c–e for illustration purpose. If we consider maximum 5 iterations as stopping criteria then the acceptable region of approximation for Taylor polynomial at development point (0, 0) is $x \in [-0.19921875, 0.19921875]$ and $y \in [-0.425, 0.425]$.

The length of traversal in positive and negative direction is kept same in the present algorithm to simplify the strategy of selection of development points which is presented in next section. Furthermore, the splitting of limiting lengths by two is chosen to get convergence quicker with iterations and to avoid stalling of algorithm in singularity zones.

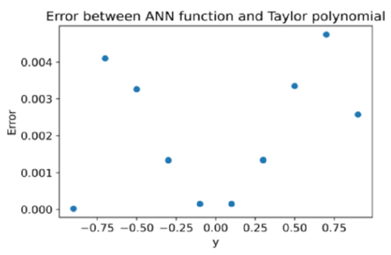
The algorithm is also applied to 5 variable dataset used in literature [33].Generally, most of the ANN regression models in science and engineering applications have input variables less than 10 and in most cases there are less than 5. The input range of each of the 5 variables is taken as [0, 1] and relationship between target output and the inputs is given by $t = 10sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5$.1200 point are sampled from input space using latin hypercube sampling and output values for these inputs points. The same 3 layer architecture with tanh function is used to train the data. The proposed algorithm will be used to find the region of acceptable approximation for 3rd degree at development point [0.5, 0.5, 0.5, 0.5, 0.5] with threshold error as 0.01. The initial lengths around the development point which are within the threshold error in the 5 variables direction is computed as $xlim_1 = 0.12, xlim_2 = 0.15, xlim_3 = 0.15, xlim_4 = 0.3, xlim_5 = 0.5$. The initial region for investigation is taken as $x_1 \epsilon$ [0.38, 0.62], $x_2 \epsilon$ [0.35, 0.65], $x_3 \epsilon$ [0.35, 0.65], $x_4 \epsilon$ [0.2, 0.8], $x_5 \epsilon$ [0, 1.0]. In each iteration, uniform grid of 3125 points (5 points in each input direction) is used to evaluate the acceptable region of approximation. The region evaluated in each iteration along with maximum error is shown in Table 1. The region of acceptable approximation considering convergence criteria as 5 maximum iterations is given as $x_1 \epsilon$ [0.455, 0.545], $x_2 \epsilon$[0.44375, 0.55625], $x_3 \epsilon$[0.44375, 0.55625], $x_4 \epsilon$[0.3875,0.6125], $x_5 \epsilon$ [0.3125,0.6875].

Strategy for selection of development points

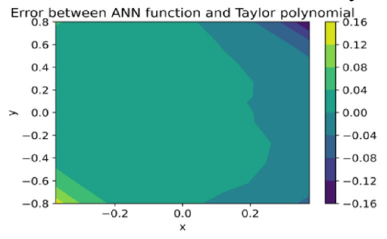
This section details a strategy for selection of development points to approximate a given input space with region wise Taylor polynomials. The algorithm presented here is geared towards minimizing the total number of iterations and error between ANN function and Taylor polynomial and is not concerned with the number of regional Taylor polynomials required to approximate the input space. The inputs to algorithm are input space to be approximated, threshold error, criteria for minimum acceptable region below which the region is to be



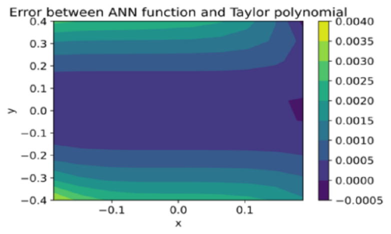
a. Error as a function of x with y=0.0



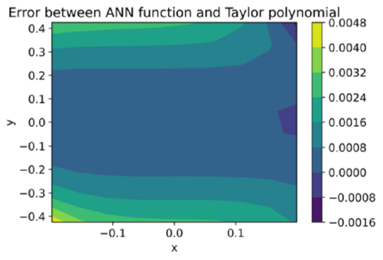
b. Error as a function of y with x=0.0



c. Error plot for $x \in [-0.375,0.375]$ and $y \in [-0.8,0.8]$



d. Error plot for $x \in [-0.1875,0.1875]$ and $y \in [-0.4,0.4]$ (1st iteration)



e. Error plot for $x \in [-0.19921875,0.19921875]$ and $y \in [-0.425,0.425]$ (5th iteration)

Fig. 9 Illustration of algorithm for computation of acceptable region of approximation for two variable function

considered as a singularity, minimum and maximum value of degree for Taylor polynomial and number of sample to be drawn for initial step. The proposed algorithm is detailed below.

Table 1 Regions for 5 variables function for various iterations

Iteration	Region	Maximum error between ANN function and Taylor polynomial
Initial	$x_1 \in$ [0.38,0.62], $x_2 \in$ [0.35,0.65], $x_3 \in$ [0.35,0.65], $x_4 \in$ [0.2,0.8], $x_5 \in$ [0.1,0.9]	0.3300
1st	$x_1 \in$ [0.44,0.56], $x_2 \in$ [0.425,0.575], $x_3 \in$ [0.425,0.575], $x_4 \in$ [0.35,0.65], $x_5 \in$ [0.25,0.75]	0.0170
2nd	$x_1 \in$ [0.47,0.53], $x_2 \in$ [0.4625,0.5375], $x_3 \in$ [0.4625,0.5375], $x_4 \in$ [0.425,0.575], $x_5 \in$ [0.375,0.625]	0.0022
3rd	$x_1 \in$ [0.455,0.545], $x_2 \in$ [0.44375,0.55625],$x_3 \in$ [0.44375,0.55625], $x_4 \in$ [0.3875,0.6125], $x_5 \in$ [0.3125,0.6875]	0.0081
4th	$x_1 \in$ [0.4475,0.5525], $x_2 \in$ [0.434375,0.565625], $x_3 \in$ [0.434375,0.565625], $x_4 \in$ [0.36875,0.63125], $x_5 \in$ [0.28125,0.71875]	0.0140
5th	$x_1 \in$ [0.45125,0.54875],$x_2 \in$ [0.4390625,0.5609375],$x_3 \in$ [0.4390625,0.5609375],$x_4 \in$ [0.378125,0.612875], $x_5 \in$ [0.296875,0.703125]	0.0120

```
Algorithm:
Step 0: Check all inputs for consistency. Let the input space to be approximated be given as $x_1 \epsilon [x_{1i}, x_{1f}], x_2 \epsilon [x_{2i}, x_{2f}], x_3 \epsilon [x_{3i}, x_{3f}], \dots\dots\dots\dots x_k \epsilon [x_{ki}, x_{kf}]$
Step 1: Select n samples of developments points inside the input space based on either latin hyper cube sampling or random sampling or any other sampling method. Let each development point be $\{x_{1sd}, x_{2sd}, x_{3sd}, \dots\dots\dots x_{ksd}\}$ where s is from 1 to n.
Step 2: Repeat step 3-4 for value of Taylor degree of polynomial from minimum to maximum
Step 3: Compute the acceptable regions of approximation for all the sample points. Remove the sample which may be a singularity regions based on criteria for minimum acceptable region. Let an acceptable region for a sample be represented as $x_1 \epsilon [x_{1sd} - \Delta xc_{1s}, x_{1sd} + \Delta xc_{1s}], x_2 \epsilon [x_{2sd} - \Delta xc_{2s}, x_{2sd} + \Delta xc_{2s}], x_3 \epsilon [x_{3sd} - \Delta xc_{3s}, x_{3sd} + \Delta xc_{3s}], \dots\dots\dots\dots x_k \epsilon [x_{ksd} - \Delta xc_{ks}, x_{ksd} + \Delta xc_{ks}]$
Step 4: Compute mean $\{\mu_1, \mu_2 \dots\dots \mu_k\}$ and variance $\{\sigma_1^2, \sigma_2^2 \dots\dots \sigma_k^2\}$ of traversal length for filtered sample development points (nf) in each input direction i.e. $\mu_i = \frac{1}{nf}\sum_{j=1}^{nf} \Delta xc_{ij}$ and $\sigma_i^2 = \frac{1}{nf}\sum_{j=1}^{nf} (\Delta xc_{ij} - \mu_i)^2$. Compute the vector $\gamma = \{\mu_1 - \sigma_1, \mu_2 - \sigma_2 \dots\dots\dots\dots, \mu_k - \sigma_k\}$.
Step 5: Select the degree of Taylor polynomial as the degree with maximum Euclidean norm of the $\gamma$ vector i.e. $\sqrt{(\mu_1 - \sigma_1)^2 + (\mu_2 - \sigma_2)^2 \dots\dots\dots\dots + (\mu_k - \sigma_k)^2}$
Step 6: Repeat step 7 for $l$=1 to k (number of dimensions of input data)
Step 7: Compute sp$_l$=ceil($\frac{x_{lf} - x_{li}}{2(\mu_l - \sigma_l)}$) (round up to the nearest integer) and divide the length into $x_1 \epsilon [x_{li}, x_{lf}]$ into sp segments i.e. $x_1 \epsilon [x_{li}, x_{li} + \frac{(x_{lf} - x_{li})}{sp_l}], x_1 \epsilon [x_{li} + \frac{(x_{lf} - x_{li})}{sp_l}, x_{li} + 2 * \frac{(x_{lf} - x_{li})}{sp_l}], x_1 \epsilon [x_{li} + 2\frac{(x_{lf} - x_{li})}{sp_l}, x_{li} + 3 * \frac{(x_{lf} - x_{li})}{sp_l}], \dots\dots\dots\dots x_1 \epsilon [x_{li} + (sp_l - 1) * \frac{(x_{lf} - x_{li})}{sp_l}, x_{li} + x_{lf}]$. The centre of each of these segments $(x_{li} + \frac{(x_{lf} - x_{li})}{2 * sp_l}, x_{li} + \frac{3 * (x_{lf} - x_{li})}{2 * sp_l}, x_{li} + \frac{5 * (x_{lf} - x_{li})}{2 * sp_l}, \dots\dots x_{li} + (2sp_l - 1)\frac{(x_{lf} - x_{li})}{2 * sp_l})$ is taken as a co-ordinate for development point
Step 8: Generate a full factorial of co-ordinates in individual directions to form total n$_t$=$sp_1 * sp_2 * sp_3 \dots\dots sp_k$ development points
Step 9: Repeat step 10 -11 for m=1 to n$_t$ (number of development point)
Step 10: Each development points can be represented by $\{x_{1i} + (2d_1 - 1)\frac{(x_{1f} - x_{1i})}{2 * sp_1}, x_{2i} + (2d_2 - 1)\frac{(x_{2f} - x_{2i})}{2 * sp_2}, x_{3i} + (2d_3 - 1)\frac{(x_{3f} - x_{3i})}{2 * sp_3}, \dots\dots\dots\dots\dots\dots x_{ki} + (2d_k - 1)\frac{(x_{kf} - x_{ki})}{2 * sp_k}\}$ where d$_1$,d$_2$,d$_3$, …… d$_k$ vary from 1 to $sp_1, sp_2, sp_3 \dots\dots sp_k$ respectively. The expected acceptable region of approximation for each development point is given as $x_1 \epsilon [x_{1i} + (d_1 - 1)\frac{(x_{1f} - x_{1i})}{sp_1}, x_{li} + x_{1i} + (d_1)\frac{(x_{1f} - x_{1i})}{sp_1}\frac{(x_{lf} - x_{li})}{sp_l}], x_2 \epsilon [x_{2i} + (d_2 - 1)\frac{(x_{2f} - x_{2i})}{sp_2}, x_{2i} + (d_2)\frac{(x_{2f} - x_{2i})}{sp_2}], x_3 \epsilon [x_{3i} + (d_3 - 1)\frac{(x_{3f} - x_{3i})}{sp_3}, x_{3i} + (d_3)\frac{(x_{3f} - x_{3i})}{sp_3}], \dots x_k \epsilon [x_{ki} + (d_k - 1)\frac{(x_{kf} - x_{ki})}{sp_k}, x_{ki} + (d_k - 1)\frac{(x_{kf} - x_{ki})}{sp_k}]$. Compute the error between ANN function and the Taylor polynomial in the expected acceptable region of approximation of the development point.
Step 11: If the computed error is less than the threshold error then output acceptable region of approximation for the development point as the expected value and go to next iteration if final iteration go to step 12
Else compute the acceptable region of approximation of the development point
    If the computed region is smaller than criteria for minimum acceptable region then consider region as singularity and go to step 12
    Else if computed acceptable region of approximation is smaller than criteria for minimum acceptable region the go to step 0 with input space to be approximated as expected acceptable region of approximation
Step 12: Check if whole input sub-space is enveloped if so output Taylor polynomial for each region.
```

The basic idea behind the algorithm is to sample some development points in the input space in order to estimate the average region of approximation and use this average region of approximation as expected region of approximation to distributed development points uniformly across the input space. For these uniformly distributed development points if the actual region of approximation is larger than the average region of approximation then for these development points the region of approximation can be

Table 2 Regions of acceptable approximation for 2nd degree Taylor polynomial

Development points	Δxc_{a1}	Δxc_{a2}	Δxc_{a3}	Δxc_{a4}	Δxc_{a5}
Point A	0.037500	0.037500	0.046875	0.046875	0.084375
Point B	0.032812	0.032812	0.032812	0.065625	0.109375
Point C	0.028125	0.028125	0.036562	0.079062	0.126562
Mean	0.328123	0.328123	0.038749	0.063854	0.106770
Std deviation	0.003827	0.003827	0.005945	0.013199	0.017320
$\mu - \sigma(\gamma)$	0.324296	0.324296	0.032804	0.050655	0.08945

Table 3 Regions of acceptable approximation for 3rd degree Taylor polynomial

Development Points	Δxc_{a1}	Δxc_{a2}	Δxc_{a3}	Δxc_{a4}	Δxc_{a5}
Point A	0.051562	0.061875	0.051562	0.10312	0.154687
Point B	0.045000	0.056250	0.056250	0.11250	0.187500
Point C	0.048750	0.048750	0.056250	0.15000	0.168750
Mean	0.048437	0.055625	0.054687	0.12187	0.170312
Std deviation	0.002688	0.005375	0.002209	0.02025	0.013441
$\mu - \sigma(\gamma)$	0.045749	0.050087	0.052478	0.10162	0.156871

taken as average region of approximation. For those development points whose actual region of approximation is smaller than the average region of approximation the average region of approximation is divided into further sub-domains. This sub division process is done by considering the average region of approximation around the failed development point as the region to be approximated and samples points can be taken in this region and the whole steps for this region is repeated. Using mean of sampled development points region of approximation, it may be reasonably to expect around half of the development points in the input space to have actual region of approximation larger than average region of approximation so in order increases the number points having larger than expected region of approximation, mean minus standard deviation of region of approximation of sampled development points is used.

The algorithm will be illustrated on 5 variable data set to approximate the ANN function in input space of $x_1 \epsilon$ [0.4, 0.6], $x_2 \epsilon$ [0.4, 0.6], $x_3 \epsilon$ [0.4, 0.6], $x_4 \epsilon$ [0.4, 0.6], $x_5 \epsilon$ [0.4, 0.6]. The minimum and maximum degree of Taylor polynomial is taken as 2 and 3, respectively. The threshold error is taken as 0.01 and stopping criteria for acceptable region of approximation is taken as 5 maximum iterations. The criterion for singularity is any acceptable traversal length being less than 0.01. Three samples developments points were taken as [0.45, 05.0.45, 0.45, 0.45] (point A), [0.50, 0.50.0.50, 0.50, 0.50] (point B), and [0.55, 0.55.0.55, 0.55, 0.55] (point C). The acceptable region of approximation for these points for 2nd and 3rd degree Taylor polynomial is presented in Tables 2 and 3. The detailed computation for point B using 3rd degree Taylor is already presented in preceding section. It can be clearly seen from Tables 2 and 3 that the norm of γ vector is larger for 3rd degree Taylor polynomial as compared to a 2nd Taylor polynomial and hence degree of Taylor polynomial to approximate ANN function is taken as 3. The computed number of segment in each direction can be given as{sp_1,sp_2,

sp_3,sp_4,$sp_5\} = \mathrm{ceil}\{\frac{0.2}{2*0.045749}, \frac{0.2}{2*0.050087}, \frac{0.2}{2*0.052478}, \frac{0.2}{2*0.10162}, \frac{0.2}{2*0.156871}\} = \mathrm{ceil}\{2.1858,1.9965,1.9055,0.9840,0.6374\} = \{3,2,2,1,1\}$. Hence the total input space will be divided into 12 (3*2*2*1*1) regions with 12 developments points. The co-ordinates of development points along with expected acceptable region of approximation is given in Table 4. It can be observed from Table 4 that the error in an expected acceptable region of approximation is less than threshold error for all but one sub region. For this region, the region of approximation would be computed by taking this region as baseline input space in next iteration.

Application of developed algorithm

In the preceding sections, the methodology to develop physics consistent ANN using inferencing algorithm is detailed. In, this section, the application of developed methodology has been illustrated on different datasets. Four benchmarking studies are provided which highlight different aspects of developed algorithm. In the first study, the difference between developed algorithm and LIME algorithm are detailed. The second study shows the singularity detection capabilities of the developed algorithm. The third and fourth studies show an example of interpretation of physics consistency using the developed algorithm and development of physics based loss function to obtain physics consistent ANN model.

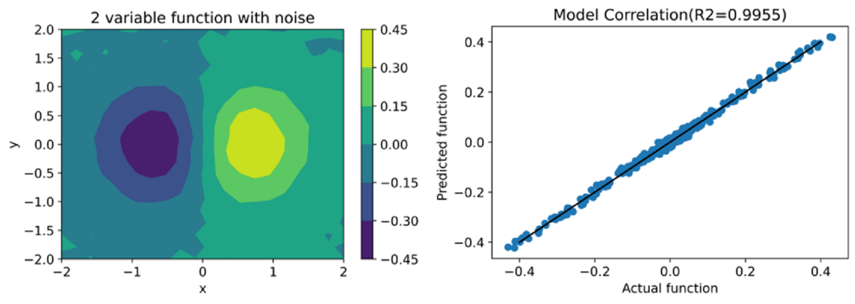
Comparison with LIME

In this case study, the results from LIME algorithm will be compared with proposed methodology. An ANN model with 3 layers and 20 neurons each was built for 2-variable function ($f = x * \exp(-(x^2 + y^2))$) which was modified by adding of Gaussian noise of zero mean and 0.01 standard deviation of 0.01. The addition of noise was done to simulate real world dataset behaviour. The modified 2-variable function along with ANN model performance is show in Fig. 10. As mentioned before in the introduction LIME builds a local model over the ANN model by sampling input space in the vicinity of point of interest whereas the current methodology characterises the ANN model

Table 4 Development points and expected region of approximation

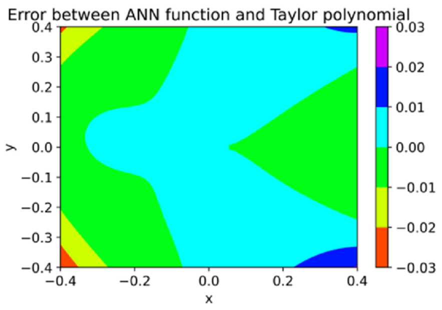
Development point	Expected region of approximation	Maximum absolute error
[0.4333, 0.45, 0.45, 0.5, 0.5]	$x_1 \in$ [0.4, 0.4667], $x_2 \in$ [0.4, 0.5], $x_3 \in$ [0.4, 0.5], $x_4 \in$ [0.4,0.6], $x_5 \in$ [0.4,0.6]	0.003518
[0.4333, 0.45, 0.55, 0.5, 0.5]	$x_1 \in$ [0.4, 0.4667], $x_2 \in$ [0.4, 0.5], $x_3 \in$ [0.5, 0.6], $x_4 \in$ [0.4,0.6], $x_5 \in$ [0.4,0.6]	0.003519
[0.4333, 0.55, 0.45, 0.5, 0.5]	$x_1 \in$ [0.4, 0.4667], $x_2 \in$ [0.5, 0.6], $x_3 \in$ [0.4, 0.5], $x_4 \in$ [0.4,0.6], $x_5 \in$ [0.4,0.6]	0.004026
[0.4333, 0.55, 0.55, 0.5, 0.5]	$x_1 \in$ [0.4, 0.4667], $x_2 \in$ [0.5, 0.6], $x_3 \in$ [0.5, 0.6], $x_4 \in$ [0.4,0.6], $x_5 \in$ [0.4,0.6]	0.002399
[0.5, 0.45, 0.45, 0.5, 0.5]	$x_1 \in$ [0.4667,0.5333], $x_2 \in$ [0.4, 0.5], $x_3 \in$ [0.4, 0.5], $x_4 \in$ [0.4,0.6], $x_5 \in$ [0.4,0.6]	0.003433
[0.5, 0.45, 0.55, 0.5, 0.5]	$x_1 \in$ [0.4667,0.5333], $x_2 \in$ [0.4, 0.5], $x_3 \in$ [0.5, 0.6], $x_4 \in$ [0.4,0.6], $x_5 \in$ [0.4,0.6]	0.007536
[0.5, 0.55, 0.45, 0.5, 0.5]	$x_1 \in$ [0.4667,0.5333], $x_2 \in$ [0.5, 0.6], $x_3 \in$ [0.4, 0.5], $x_4 \in$ [0.4,0.6], $x_5 \in$ [0.4,0.6]	0.010849*
[0.5, 0.55, 0.55, 0.5, 0.5]	$x_1 \in$ [0.4667,0.5333], $x_2 \in$ [0.5, 0.6], $x_3 \in$ [0.5, 0.6], $x_4 \in$ [0.4,0.6], $x_5 \in$ [0.4,0.6]	0.009431
[0.5667, 0.45, 0.45, 0.5, 0.5]	$x_1 \in$ [0.5333,0.6], $x_2 \in$ [0.4, 0.5], $x_3 \in$ [0.4, 0.5], $x_4 \in$ [0.4,0.6], $x_5 \in$ [0.4,0.6]	0.004257
[0.5667, 0.45, 0.55, 0.5, 0.5]	$x_1 \in$ [0.5333,0.6], $x_2 \in$ [0.4, 0.5], $x_3 \in$ [0.5, 0.6], $x_4 \in$ [0.4,0.6], $x_5 \in$ [0.4,0.6]	0.003938
[0.5667, 0.55, 0.45, 0.5, 0.5]	$x_1 \in$ [0.5333,0.6], $x_2 \in$ [0.5, 0.6], $x_3 \in$ [0.4, 0.5], $x_4 \in$ [0.4,0.6], $x_5 \in$ [0.4,0.6]	0.005373
[0.5667, 0.55, 0.55, 0.5, 0.5]	$x_1 \in$ [0.5333,0.6], $x_2 \in$ [0.5, 0.6], $x_3 \in$ [0.5, 0.6], $x_4 \in$ [0.4,0.6], $x_5 \in$ [0.4,0.6]	0.003434

locally by using its local differentials. Furthermore, the local model from LIME algorithm depends on parameters chosen i.e. number of samples and standard deviation of weightage function. Moreover, since LIME model building involves sampling, different model would be outputted for different runs of LIME algorithm. This is confirmed by using the regression tutorial provided by authors of lime package (https://github.com/marcotcr/lime). But the Taylor polynomial developed from current methodology is always unique. Below the equations of Taylor polynomial along with 4 LIME instances of similar polynomials generated at development point of (0, 0) are presented. For the 4 LIME models, the all sampling points are taken in the region of $x \in [-0.4, 0.4]$ and $y \in [-0.4, 0.4]$. It is interesting to note that the coefficient of x and y terms which are equal to ANN first order differentials at (0, 0) are close to target function differential values of 1 and 0 for all the polynomials. Also, it can be observed that for different LIME instances the coefficients values and their signs are quite different hence offer different explanations of the ANN model (for instance, value of x^3,x^2y and xy^2 coefficient and sign and value y^4 coefficient). Hence it is difficult for users to interpret and understand the physics consistency of ANN model by using LIME algorithm. It should be noted that the LIME explanation variation is even more significant in the tutorial provided by the authors but in this test case the variation was reduced due to sampling in small domain. Furthermore, the error profile between the ANN model and the 5 different polynomials in the region of $x \in [-0.4, 0.4]$ and $y \in [-0.4, 0.4]$ is computed using regular grid of 2500 points and is plotted in Fig. 11. Couple of things can be observed from error plots. Firstly, the range of error plots of Taylor polynomial and LIME 3 is similar but coefficients of some terms are vastly different. More importantly, it can be observed the Taylor polynomial error is less and is of one sign in the neighbourhood of development point which is desirable characteristics but this is not true for the LIME polynomial. This due to fact Taylor polynomial characterises the ANN model whereas LIME algorithm built a model on top of ANN. Moreover, using the current methodology Taylor polynomials with any user specified accuracy of ANN model can be built which is not possible with LIME. From a theoretical point of view the proposed Taylor polynomial can captures the behaviour of ANN model more efficiently than LIME algorithm. In summary, the major differences between proposed methodology and LIME model are (i) LIME built a local model in neighbourhood of development point whereas the proposed methodology develops a Taylor polynomial using local differentials which characteristics the ANN

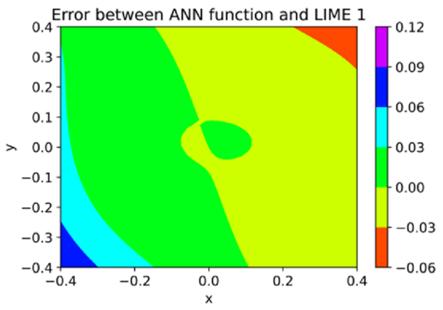


a. Two variable output function with noise b. ANN actual vs predicted plot

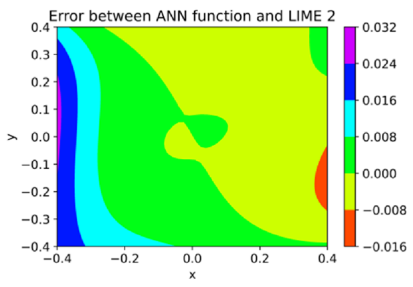
Fig. 10 Two variable output with noise and ANN model performance



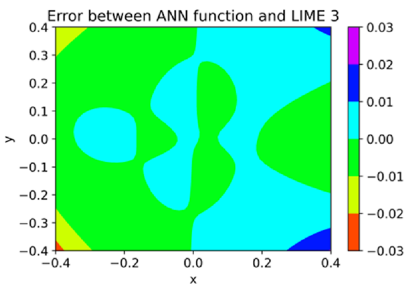
a. Error between ANN and Taylor polynomial



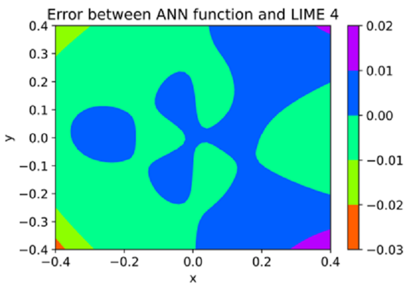
b. Error between ANN and LIME1



c. Error between ANN and LIME2



d. Error between ANN and LIME3



e. Error between ANN and LIME4

Fig. 11 Error profile of LIME models and current methodology

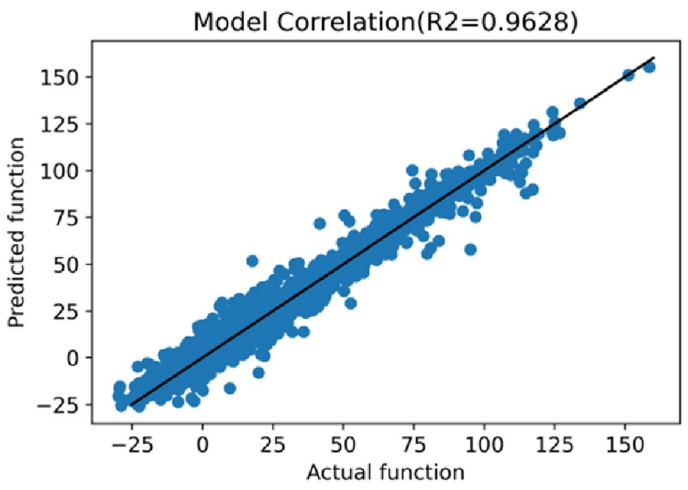


Fig. 12 Prediction vs actual for 5 variable function with singularity

model (ii) LIME model is parameter dependent whereas proposed methodology is not (iii) Taylor polynomial from current methodology is unique whereas LIME outputs different polynomial for different runs (iv) proposed methodology can be used to develop series of Taylor polynomial to approximate ANN model within a given error limit which is not possible with LIME algorithm.

Taylor polynomial:

$$\begin{aligned} \text{z} =& 0.00252 + 1.01668x - 0.008351\text{y} \\ & - 0.08410x^2 - 0.04547xy - 0.08410y^2 \\ & - 0.89973x^3 + 0.01683x^2y - 1.20296xy^2 \\ & + 0.02193y^3 + 0.57378x^4 + 0.22771x^3y \\ & + 0.03968x^2y^2 + 0.10543xy^3 + 0.00738y^4 \end{aligned}$$

LIME 1(number of samples = 100, variance = 0.05, Run 1)

$$\begin{aligned} \text{z} =& 0.00252 + 1.01349x - 0.00961y \\ & - 0.09571x^2 - 0.07320xy - 0.00879y^2 \\ & - 0.50899x^3 + 0.18727x^2y - 0.48057xy^2 \\ & + 0.18900y^3 + 0.02627x^4 - 0.00660x^3y \\ & + 0.03175x^2y^2 - 0.01568xy^3 - 0.01264y^4 \end{aligned}$$

LIME 2(number of samples = 100, variance = 0.05, Run 2)

$$\begin{aligned} \text{z} =& 0.00252 + 1.01509 - 0.00909y - 0 \\ & .08304x^2 - 0.04583xy - 0.00860y^2 \\ & - 0.68918x^3 - 0.06738x^2y - 0.96159xy^2 \\ & + 0.13194y^3 - 0.03539x^4 + 0.00637x^3y \\ & + 0.01730x^2y^2 + 0.04382xy^3 - 0.00046y^4 \end{aligned}$$

LIME 3(number of samples = 500, variance = 0.1, Run 1)

$$\begin{aligned} \text{z} =& 0.00252 + 1.01677 - 0.00838y \\ & - 0.08252x^2 - 0.04558xy - 0.00857y^2 \\ & - 0.93159x^3 + 0.02465x^2y - 1.14162xy^2 \\ & + 0.02046y^3 + 0.44982x^4 + 0.192684x^3y \\ & + 0.01684x^2y^2 + 0.11463xy^3 + 0.00746y^4 \end{aligned}$$

LIME 4(number of samples = 500, variance = 0.1, Run 2)

$$
\begin{aligned}
\text{z} =&0.00252 + 1.01668 - 0.00837y \\
&- 0.08259x^2 - 0.04577xy - 0.00881y^2 \\
&- 0.92999x^3 + 0.01952x^2y - 1.14160xy^2 \\
&+ 0.02128y^3 + 0.44339x^4 + 0.17175x^3y \\
&+ 0.04656x^2y^2 + 0.12096xy^3 + 0.01768y^4
\end{aligned}
$$

Singularity detection

One of the proposed usages of the proposed methodology is for detection of singularity zones in ANN prediction space. As seen from Sect. 3.3, Taylor polynomial has very minimal radius of convergence when the development point is near a singularity. Making use of this property, the singularity can be detected if the region of approximation of development point is less than threshold value. The threshold can be selected on heuristic judgement and previous experience. In a section, a subpart to proposed methodology is used to detect singularity. It is difficult from real world datasets to ascertain whether a region is actual singularity or not and hence an artificial dataset will used to confirm the accuracy of proposed methodology. 5 variable function which was used in Sect. 4.2 will be slightly modified to have singularity at x_4=0.5 and x_5=0.5 i.e. $t = 10sin(\pi x_1x_2) + 20(x_3 - 0.5)^2 + 10x_4 + \frac{4}{(x_5-0.5)^2+(x_4-0.5)^2+0.05}$. 2500 input points were generated using latin hypercube sampling. An ANN model of 3 layers with 15 neurons was built using the data. The performance of the developed ANN model is shown in Fig. 12. Two developed points are taken as point A [0.49, 0.49, 0.49, 0.49, 0.49] which is in close proximity to the singularity and point B [0.80, 0.80, 0.80, 0.80, 0.80] which is far away from singularity. Using the methodology in Sect. 4.1 the region of approximation is computed for both development points with maximum error of 0.01 and stopping criteria of 5 iterations. The initial lengths around the development point A which are within the threshold error in the 5 variables direction is computed as $xlim_1 = 0.06, xlim_2 = 0.08, xlim_3 = 0.06, xlim_4 = 0.03, xlim_5 = 0.05$.

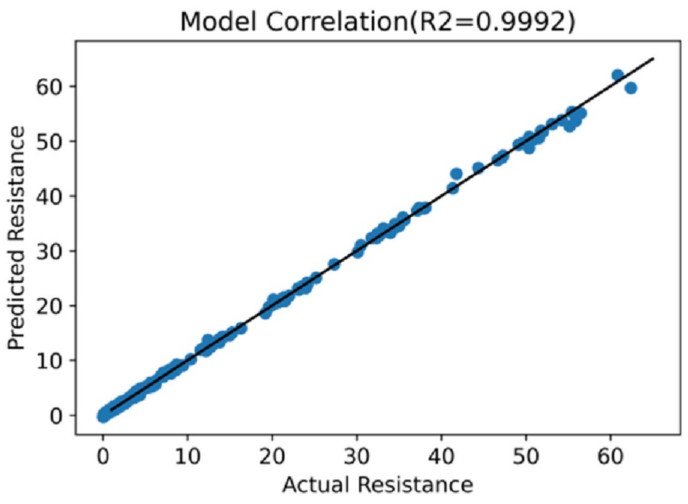


Fig. 13 Actual vs predicted for the yatch hydrodynamics ANN model

Similar for point B the initial lengths are computed as $xlim_1 = 0.08, xlim_2 = 0.15, xlim_3 = 0.1, xlim_4 = 0.08, xlim_5 = 0.1$. The computed region of approximation for point A and B are $x_1\epsilon$[0.485312,0.4946875], $x_2\epsilon$[0.48375,0.49625], $x_3\epsilon$[0.4853125,0.4946875], $x_4\epsilon$ [0.48765625,0.49234375], $x_5\epsilon$ [0.48609375,0.49390625] and $x_1\epsilon$[0.79,0.81], $x_2\epsilon$[0.78125,0.81875], $x_3\epsilon$[0.7875,0.8125], $x_4\epsilon$ [0.79,0.81], $x_5\epsilon$ [0.78125,0.81875] respectively. It can be observed that the region of approximation is significantly smaller for development point A as compared to development point B especially in x_4 and x_5 directions therefore one can infer that singularity exists near to development point A in x_4 and x_5 directions. Hence, it can be seen that present methodology can help locate singular zones in ANN prediction space. It should also be noted the methodology only detects singularity in ANN prediction and not the actual output space.

Interpretation of physics consistency

For the next 2 studies, datasets were chosen from UCI repository maintained by the center for machine learning and intelligent systems at the University of California [34]. The first dataset will be used to illustrate the usage of the inferencing algorithm to obtain input output relations of ANN model which can be assessed for physics consistency. The first data set is the yatch hydrodynamic dataset generated using full scale experiments performed at delft ship hydromechanics laboratory using 22 different hull forms [35]. The data set contains 308 data points with 6 input variables i.e. longitudinal position of the center of buoyancy, prismatic coefficient, length-displacement ratio, beam-draught ratio, length-beam ratio, and Froude number and one output variable which is the residuary resistance of the ship hull. In first step, the regression model was developed from this. An ANN with 3 hidden layer each having 20 neurons was developed for this data. The ANN model prediction of residuary resistance as compared to actual resistance is shown in Fig. 13.

In next step, the inferencing algorithm would be applied to total input space or important sub-spaces. For the sake of brevity, the inferencing algorithm is illustrated for a single input sub-space: centre of buoyancy(x_1) ϵ [−2.4, − 2.6], prismatic coefficient (x_2) ϵ [0.545, − 0.55], length-displacement ratio (x_3) ϵ [4.75, 4.79], beam-draught ratio (x_4) ϵ [3, 3.06], length-beam ratio (x_5) ϵ [3, 3.06], Froude number(x_6) ϵ [0.418, 0.42]. Two random points A [− 2.5, 0.55, 4.75, 3, 3, 0.42] and B [− 2.4, 0.545, 4.77, 3, 3, 0.418] were sampled

Table 5 Development points and expected region of approximation (yatch hydrodynamics)

Development point	Expected region of approximation	Maximum absolute error
[− 2.45, 0.5475, 4.77, 3.03, 3.03, 0.419] Region A	$x_1\epsilon$ [− 2.4, − 2.5], $x_2\epsilon$ [0.545, 0.55], $x_3\epsilon$ [4.75, 4.79], $x_4\epsilon$ [3,0.6], $x_5\epsilon$ [3,3.06], $x_6\epsilon$ [0.418,0.42]	0.006311
[− 2.55, 0.5475, 4.77, 3.03, 3.03, 0.419] Region B	$x_1\epsilon$ [− 2.5, − 2.6], $x_2\epsilon$ [0.545, 0.55], $x_3\epsilon$ [4.75, 4.79], $x_4\epsilon$ [3, 0.6], $x_5\epsilon$ [3, 3.06], $x_6\epsilon$ [0.418, 0.42]	0.006291

in the input sub space. The region of approximation for an 2nd degree Taylor polynomial with threshold error of 0.1 was computed using algorithm in Sect. 4.1. The values of region of acceptable approximation is not presented for the sake of brevity. Based on the average minus standard deviation values of acceptable length of approximation two development points in the input sub-space was selected. As seen from Table 5 the absolute error in both the expected region of approximation is less than the threshold error and hence the Taylor polynomial with initial development point can be used to approximate these regions.

The Taylor polynomial for only region A is given below for the sake of brevity.

Region A:

$$\begin{aligned}\frac{y-10.49535}{15.13585} =\ & 2.25929+0.12178x_{a1}-0.08827x_{a2}\\ &+0.26720x_{a3}-0.45277x_{a4}-0.58052x_{a5}\\ &+4.92001x_{a6}+0.20525\frac{x^2{}_{a1}}{2}-0.12660\frac{x^2{}_{a2}}{2}\\ &-0.04975\frac{x^2{}_{a3}}{2}+0.03043\frac{x^2{}_{a4}}{2}+0.05494\frac{x^2{}_{a5}}{2}\\ &+1.80831\frac{x^2{}_{a6}}{2}-0.06432x_{a1}x_{a2}-0.02936x_{a1}x_{a3}\\ &-0.03044x_{a1}x_{a4}-0.04831x_{a1}x_{a5}\\ &+0.13864x_{a1}x_{a6}+0.07139x_{a2}x_{a3}-0.03906x_{a2}x_{a4}\\ &+0.00632x_{a2}x_{a5}+0.17492x_{a2}x_{a6}+0.01781x_{a3}x_{a4}\\ &+0.01473x_{a3}x_{a5}-0.11590x_{a3}x_{a6}\\ &+0.04343x_{a4}x_{a5}-0.22305x_{a4}x_{a6}-0.34357x_{a5}x_{a6}\end{aligned}$$

where

$x_{a1} = \frac{(x_1+2.45)}{1.51076}, x_{a2} = \frac{(x_2-0.5475)}{0.023252}, x_{a3} = \frac{(x_3-4.77)}{0.25264}, x_{a4} = \frac{(x_4-3.03)}{0.54730}, x_{a5} = \frac{(x_5-3.03)}{0.24795}, x_{a6} = \frac{(x_6-0.419)}{0.10077}$

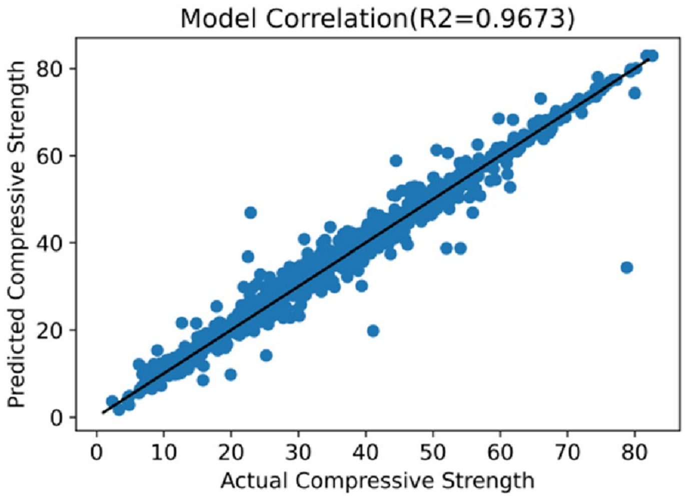


Fig. 14 Actual vs predicted for the concrete compressive strength ANN model (initial model)

Some of the major relationship explanations inferred between input and output variables using Taylor polynomial in region A are:

- Residual resistance is directly proportional to longitudinal position of the center of buoyancy, length-displacement ratio, and Froude number and inversely proportional to prismatic coefficient, beam-draught ratio, and length-beam ratio.
- Froude number has largest influence of residual resistance.
- Residual resistance is directly proportional to length-displacement ratio values but is inversely proportional to the square value of length-displacement ratio value.
- Residual resistance is inversely proportional to beam-draught ratio values but is directly proportional to the square value of beam-draught ratio value
- The interaction term of the prismatic coefficient and Froude number is third largest contributor to residual resistance and is a positive iteration term i.e. decrease/increase of both variables will increase residual resistance but decrease of one variable and increase of other variable with decrease residual resistance

From the developed relationships, the physics consistency can be assessed with the known physics knowledge. The first two observations seem to physics consistent from available open literature [35] whereas the third to fifth observations physics consistency can be assessed by domain expert. Similarly, inference algorithm can be applied to other regions and additional points can be generated for region where physics inconsistency is found. The process can be iterated till all the regions in ANN regression model satisfy the physics consistency.

Physics based differential loss function

The second engineering dataset which will be used to illustrate the physics based loss function is the concrete compressive strength dataset [34].The dataset consists of concrete compressive strength for different values mixture constituents along with the age

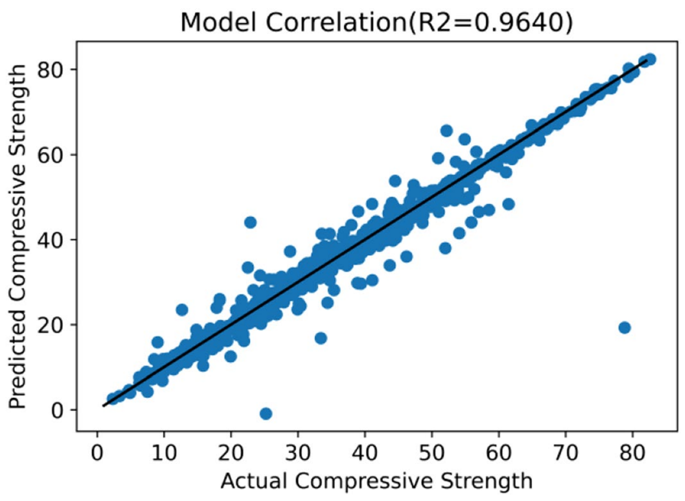


Fig. 15 Actual vs predicted for the concrete compressive strength ANN model (Model with physics loss function for superplasticizer)

of the concrete. The seven constituents considered in the dataset are cement, blast furnace slag, fly ash, water, superplasticizer, coarse and fine aggregates. An ANN model with 3 hidden layer each having 20 neurons was developed for this data with 70–15–15 train, test, validation data split. The ANN model prediction of residuary resistance as compared to actual resistance is shown in Fig. 14.

Since the objective of this test case is only to illustrate the physic based loss function, only a simple first degree Taylor polynomial with development point at the mean of the data will be developed. The input value of development point is cement (x_1):281.17, blast furnace slag (x_2):73.89, fly ash (x_3):54.19, water (x_4):181.57, superplasticizer (x_5):6.2, coarse aggregate (x_6):972.92, fine aggregate(x_7):773.58 and age (x_8):45.66. The first degree Taylor polynomial for the above development points is given below:

$$\begin{aligned}\frac{(y-35.81796)}{16.69763} =\, &1.02291 + 0.13811x_{c1} + 0.29268x_{c2}\\ &+ 0.12332x_{c3} - 1.03798x_{c4} - 0.60545x_{c5}\\ &- 0.49203x_{c6} + 1.17147x_{c7} - 0.68021x_{c8}\end{aligned}$$

where
$x_{c1} = \frac{x_1-281.17}{104.45562}, x_{c2} = \frac{x_2-73.89}{86.23744}, x_{c3} = \frac{x_3-54.19}{63.96593}, x_{c4} = \frac{x_4-181.57}{21.34384}, x_{c5} = \frac{x_5-6.2}{5.97094}, x_{c6} =. \frac{x_6-972.92}{77.71620}, x_{c7} = \frac{x_7-773.58}{80.13705}, x_{c8} = \frac{x_8-45.66}{63.13923}$

From the literature [36, 37], it can be seen that the compressive strength (y) in is proportional to the amount of superplasticizer (x_5) which means the coefficient of x_{c5} in Taylor polynomial should be positive. But that is not the case which means this physics relationship is violated by the ANN regression model near the mean. One way to make the ANN model consistent with physics is to add more data points near mean of data and re-train the model. This is the preferred method. However, in cases where addition of more data is not possible, physics based loss can be added to total loss. This physics loss can be taken as an average of differential of output w.r.t. violated quantity over all the inputs points. Each constant value in the Taylor polynomial generated by inferencing algorithm represents a differential quantity and the differential quantity of the violated term can be added in physics loss function. If the physics relationship is directly proportional relation then negative differential should be used and positive differential should be used inversely proportional relationship.

$$\text{Loss} = \text{MSE loss} + \text{Regularization loss} + \Omega * \text{physics loss},$$

where Ω is multiplier of physics loss (0.1 in present case)

$$\text{Physicsloss(superplasticizer)} = -\frac{1}{n}\sum_{i=1}^{n}\frac{dy}{dx_5}(x_5 = x_{5i}),$$

where n is number of data points.

The model is re-trained using the new loss function and plot for actual vs predicted is shown in Fig. 15. Again the inference algorithm is used with mean as the development point and developed the relationship shows that the ANN regression model is consistent with physics with regard to superplasticizer relationship. The relationship between compressive strength and fly ash has changed in new ANN regression model and this should also be check by user for physics consistency based on available literature or expert onion.

$$\begin{aligned}\frac{(y-35.81796)}{16.69763} =& 1.01053+1.50332x_{c1}+0.95170x_{c2}\\ &-1.41150x_{c3}-0.13374x_{c4}+0.54586x_{c5}\\ &-0.16829x_{c6}+3.092357x_{c7}-0.63694x_{c8}.\end{aligned}$$

Conclusions

The present manuscript presents a method to develop physics consistent ANN regression models in engineering and science applications. The developed method assesses the physics consistency of ANN model in different input sub-space regions and regions where physics inconsistency is found, additional points in that specific region are added and ANN model is re-trained. In cases where addition of points is not possible or addition of points isn't sufficient, a physics based loss can be used in the training of the ANN model. The assessment of physics consistency of the ANN model is done using an inferencing algorithm which interprets input output relationship of ANN regression model. The inferencing algorithm is based on Taylor polynomial. Taylor polynomial is a polynomial computed using the ANN function value and ANN gradients at an input point called development points. However, Taylor polynomial is only accurate if the distance between estimate point and development point is less than the radius of convergence. Hence in this paper region-wise Taylor polynomials are used which approximates the ANN function.

The paper also studies basic properties of gradients of ANN function and effect of degree of Taylor polynomial on the approximation capability of the Taylor polynomial. The algorithm is split into two sections. The first section of the algorithm deals with computing the acceptable region of around a given development point. This algorithm estimates the initial region for investigation based on limiting traversal length in each input direction within threshold error. The initial region is compressed or expanded based on error in that region similar to a bisection algorithm until convergence criteria is met. The second section of the algorithm deals with strategy for selection of development points. Initially several development points are randomly sampled and acceptable region of approximation for all these sample point is computed using first section of algorithm. The expected acceptable region of approximation is estimated using the sampled development points. Then the input space is split into several domains based on the expected acceptable region of approximation. If error in split domain is less than threshold error than the computed Taylor polynomial is used to approximate that region. If the error in any of the regions is greater than threshold error, then the specific regions are split into several regions based on the same algorithm. The algorithms also find singular zones in ANN predictions space as the region which have very small acceptable region of approximation or the regions which cannot be approximated by a Taylor polynomial.

Furthermore, the algorithm was applied several datasets to demonstrate the process of developing the physics consistent regression model. The first case study was used to compare current methodology with LIME algorithm while second case study was used to demonstrate singularity detection. The third case study is used to illustrate the inferencing algorithm to check physics consistency and fourth case study is used to illustrate the physics based loss function. The presented methodology will help engineer and researchers develop physics consistent ANN regression models.

Acknowledgements
The author is extremely thankfully to his family who have extended their unconditional support in difficult times of author's life. The author especially wants to acknowledge his father Mr. Nuton Israel Nicodemus who is the main inspiration and motivator for this work.

Authors' information
Mr E. Rajasekhar Nicodemus is a post graduate (Master of Technology) in mechanical engineering from Indian Institute of Technology (IIT), Roorkee. Mr Nicodemus did his master dissertation at Technische Universität Darmstadt, Germany through DAAD scholarship. Mr. Nicodemus published 8 papers in various international esteemed journals and conferences. He is also a reviewer for many esteemed journals and reviewed over 45 research articles. Mr. Nicodemus worked in the automotive sector for over 9 years as Lead Engineer CAE methods in General motors technical center India. During his stint he worked on several challenging multi-physics problems in engine, transmission and turbo-charger. Mr. Nicodemus has two US patents (one issued one published). Mr. Nicodemus research interests include algorithm development, CAE, tribology, machine leaning and data analytics.

Author contributions
Since this is a sole author paper. All the work in the paper was done by ERN. All authors read and approved the final manuscript.

Funding
The author declare that there are no funding for the work.

Availability of data and materials
The datasets generated and/or analysed during the current study are available in the UCI repository, https://archive.ics.uci.edu/ml/index.php.

Declarations

Competing interests
The author declare that they have no competing interests.

Received: 2 February 2022 Accepted: 4 June 2022
Published online: 02 July 2022

References
1. Wu B, Filipi Z, Assanis DN, Kramer DM, Ohl GL, Prucka MJ, Divalentin E. Using artificial neural networks for representing the air flow rate through a 2.4 L VVT Engine,(2004) SAE International 2004-01-305 https://doi.org/10.4271/2004-01-3054.
2. Wu B, Prucka RG, Filipi Z, Kramer D M, Ohl GL. Cam-phasing optimization using artificial neural networks as surrogate models-maximizing torque output. 2005. SAE International 2005-01-3757 https://doi.org/10.4271/2005-01-3757.
3. Meyer S, Greff A. New calibration methods and control systems with artificial neural networks. (2002) SAE International 2002-01-1147 https://doi.org/10.4271/2002-01-1147.
4. Wendeker M, Czarnigowski J. Hybrid air/fuel ratio control using the adaptive estimation and neural network. (2000) SAE International 2000-01-1248 https://doi.org/10.4271/2000-01-1248.
5. Grimaldi CN, Mariani F, OBD Engine Fault Detection using a Neural Approach.(2001) SAE International 2001-01-0559 https://doi.org/10.4271/2001-01-0559.
6. Nicodemus ER, Ray S. Compound neural network architecture for stress distribution prediction. US20210174198A1. 2020. https://patents.google.com/patent/US20210174198A1/.
7. Paul S, Kapoor K, Jasani D, Dudhwewala R ,Gowda VB, Nair TRG. Application of artificial neural networks in aircraft maintenance, repair and overhaul solutions. Total Engineering, Analysis and Manufacturing Technologies. 2008
8. Altman R, Carifio J, Halverson J, Nelson BD. Estimating Calabi-Yau hypersurface and triangulation counts with equation learners. J High Energy Phys. 2019;03:186.
9. Bull K, He Y-H, Jejjala V, Mishra C. Machine learning CICY threefolds. Phys Lett B. 2018;785:65–72. https://doi.org/10.1016/j.physletb.2018.08.008.
10. Liano K. Robust error measure for supervised neural network learning with outliers. IEEE Trans Neural Netw. 1996;7(1):246–50. https://doi.org/10.1109/72.478411.
11. Karpatne A, Watkins W, Read J, Kumar V. Physics-guided neural networks (PGNN): An application in lake temperature modelling. (2017) arXiv preprint arXiv:1710.11431.

12. Jia X, Willard J, Karpatne A, Read JS, Zwart JA, Steinbach M, Kumar V. Physics-guided machine learning for scientific discovery: an application in simulating lake temperature profiles. ACM/IMS Trans Data Sci. 2021;2(3):1–26. https://doi.org/10.1145/3447814.
13. Zhang R, Liu Y, Sun H. Physics-guided convolutional neural network (PhyCNN) for data-driven seismic response modeling. Eng Struct. 2020;215: 110704. https://doi.org/10.1016/j.engstruct.2020.110704.
14. Wang J, Li Y, Zhao R, Gao RX. Physics guided neural network for machining tool wear prediction. J Manuf Syst. 2020;57:298–310. https://doi.org/10.1016/j.jmsy.2020.09.005.
15. Weinam E, Yu B. The Deep Ritz Method: a deep learning-based numerical algorithm for solving variational problems. Commun Math Stat. 2018;6:1–12. https://doi.org/10.1007/s40304-018-0127-z.
16. Raissi M, Perdikaris P, Karniadakis GE. Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J Comput Phys. 2019;378:686–707. https://doi.org/10.1016/j.jcp.2018.10.045.
17. Yu SJ, Ryu IG, Park MJ, Im JK. Long-term relationship between air and water temperatures in Lake Paldang, South Korea. Environ Eng Res. 2021;26(4):200177. https://doi.org/10.4491/eer.2020.177.
18. Simonyan K, Vedaldi A, Zisserman A. Deep inside convolutional networks: visualising image classification models and saliency maps. Workshop at International Conference on Learning Representations. 2014.
19. Zeiler MD, Fergus R. Visualizing and understanding convolutional networks. ECCV Part 1 LNCS. 2014;8689(2014):818–33. https://doi.org/10.1007/978-3-319-10590-1_53.
20. Springenberg JT, Dosovitskiy A, Brox T, Riedmiller M. Striving for simplicity: the all convolutional Net. ICLR-2015 workshop. 2014.
21. Selvaraju RR, Cogswell M, Das A, Vedantam R, Parikh D, Batra D. Grad-CAM: visual explanations from deep networks via gradient-based localization. IEEE International Conf Comput Vis (ICCV) 2017;618–626. https://doi.org/10.1109/ICCV.2017.74.
22. Bach S, Binder A, Montavon G, Klauschen F, Muller K-R, Samek W. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. PLoS ONE. 2015;10(7):e0130140.
23. Sundararajan M. Taly A, Yan Q. Axiomatic attribution for deep networks. In: Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, PMLR 70. 2017.
24. Chattopadhyay A, Manupriya P, Sarkar A, Balasubramanian VN. Neural network attributions: a causal perspective. In: Proceedings of the 36th International Conference on Machine Learning, PMLR 97; 2019;981–990.
25. Ribeiro MT, Singh S, Guestrin C. Why Should I Trust You?: Explaining the Predictions of Any Classifier. 2016; ArXiv. https://arxiv.org/abs/1602.04938.
26. Taylor B. Methodus Incrementorum Directa et Inversa, London. 1715
27. Yang J, Hu H, Potier-Ferry M. Solving large-scale problems by Taylor meshless method. Int J Numer Methods Eng. 2017;112(2):103–24. https://doi.org/10.1002/nme.5508.
28. Yang J, Hu H, Koutsawa Y, Potier-Ferry M. Taylor meshless method for solving non-linear partial differential equations. J Comput Phys. 2017;348:385–400. https://doi.org/10.1016/j.jcp.2017.07.034.
29. Yang J, Hu H, Potier-Ferry M. Least-square collocation and Lagrange multipliers for Taylor meshless method. Numer Meth Part D E. 2019;35(1):84–113. https://doi.org/10.1002/num.22287.
30. Czarnecki WM, Osindero S, Jaderberg M, Świrszcz G, Pascanu R, Sobolev Training for Neural Networks. 2017; arXiv:1706.04859.
31. Chen DS, Jain RC. A robust back propagation learning algorithm for function approximation. IEEE Trans Neural Netw. 1994;5(3):467–79. https://doi.org/10.1109/72.286917.
32. Rusiecki AL. Robust learning algorithm based on iterative least median of squares. Neural Process Lett. 2012;36:145–60. https://doi.org/10.1007/s11063-012-9227-z.
33. Papadopoulos G, Edwards PJ. Murray AF Confidence estimation methods for neural networks: a practical comparison. IEEE T Neural Networ. 2000;12(6):1278–87. https://doi.org/10.1109/72.963764.
34. https://archive.ics.uci.edu/ml/index.php.
35. Gerritsma J, Onnink R, Versluis A. Geometry, resistance and stability of the delft systematic yacht hull series. Int Shipbuild Progr. 1981;28:276–97.
36. Cheng Yeh I. Modeling of strength of high performance concrete using artificial neural networks. Cem Concr Res. 1998;2(12):1797–808.
37. Mohammed MS, Mohamed SA, Johari MAM. Influence of superplasticizer compatibility on the setting time, strength and stiffening characteristics of concrete. Adv Appl Sci. 2016;1(2):30–6.

Publisher's Note