TMPNN: High-Order Polynomial Regression Based on Taylor Map Factorization

Andrei Ivanov[1], **Stefan Ailuro**[2]

[1]Independent Researcher, Toronto, Canada
[2]Independent Researcher, Moscow, Russia
05x.andrey@gmail.com, ailuro.sm@gmail.com

Abstract

The paper presents Taylor Map Polynomial Neural Network (TMPNN), a novel form of very high-order polynomial regression, in which the same coefficients for a lower-to-moderate-order polynomial regression are iteratively reapplied so as to achieve a higher-order model without the number of coefficients to be fit exploding in the usual curse-of-dimensionality way. This method naturally implements multi-target regression and can capture internal relationships between targets. We also introduce an approach for model interpretation in the form of systems of differential equations. By benchmarking on Feynman regression, UCI, Friedman-1, and real-life industrial datasets, we demonstrate that the proposed method performs comparably to the state-of-the-art regression methods and outperforms them on specific tasks.

Introduction and Related Works

We consider the regression problem as finding the mapping f for feature vector $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n) \in \mathbb{R}^n$ of n input real-valued variables to the dependent target vector $\mathbf{Y} = (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_m) \in \mathbb{R}^m$ of m output real-valued variables:

$$f: \ (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n) \ \rightarrow \ (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_m). \qquad (1)$$

The first group of methods to solve this problem is based on the regular polynomial regression

$$\mathbf{y}_l = w_0 + \sum_i w_i \mathbf{x}_i + \sum_{i,j} w_{ij} \mathbf{x}_i \mathbf{x}_j + \ldots \qquad (2)$$

To fit model (2) efficiently and reduce risk of overfitting, most authors suggest either optimal strategies for the selection of the nonlinear terms (Davierwalla 1977; Dette 1995; Fan et al. 1997; Lewis 2007; Hofwing, Strömberg, and Tapankov 2011; Pakdemirli 2016) or different factorization schemes (Rendle 2010; Freudenthaler, Schmidt-Thieme, and Rendle 2011; Blondel et al. 2016).

The second group of methods is machine learning (ML) models like Support Vector Regression (SVR), Gaussian Process Regression (GPR), Random Forest Regression (RFR), or neural networks (NN). These black-box models were initially developed for interpolation problems. GPR with the Gaussian kernel regresses to the mean function when extrapolating far enough from training. Tree-based methods presume the output will be constant values if predictions are made outside the range of the original set. Various techniques, such as kernel-based modified GPR (Wilson and Adams 2013) or regression-enhanced RFR (Zhang, Nettleton, and Zhu 2019), are introduced to improve the extrapolation property of these models. The state-of-the-art models for regression problem (1) are CatBoost as a boosting method on decision trees (Dorogush, Ershov, and Gulin 2018), TabNet for deep NN (Arik and Pfister 2021), and DNNR for Differential Nearest Neighbors Regression (Nader, Sixt, and Landgraf 2022).

The paper introduces a new regression model based on the Taylor map factorization that can be implemented as a polynomial neural network (PNN) with shared weights. The earliest and most relevant to our research is the paper of authors (López, Huerta, and Dorronsoro 1993), where the connection between the system of ordinary differential equations (ODEs) and polynomial neural network (PNN) is introduced. Further, the PNN architectures were also widely highlighted in the literature (Oh, Pedrycz, and Park 2003). (Zjavka 2011) proposes a polynomial neural architecture approximating differential equations. The Legendre polynomials are chosen as a basis by (Yang, Hou, and Luo 2018). (Ivanov, Golovkina, and Iben 2020) suggest an algorithm for translating ODEs to PNN without training. (Wu et al. 2022) examined the extrapolation ability for PNNs in simple regression and more complicated computer vision problems. (Fronk and Petzold 2023) applied PNN for learning ODEs.

Following these works, we propose an approach for the general-purpose multi-target regression problem. In contrast to traditional techniques, where several single-output models are either used independently or chained together (Xioufis et al. 2012; Borchani et al. 2015), the proposed model processes all targets simultaneously without splitting them into different single-output models. Moreover, we contribute to the overall goal of transparent and safer machine learning models in the following ways.

- We propose a new regression model at the intersection of classical methods and deep PNN.
- The model is theoretically grounded and closely relates to the theory of ODEs, which opens up the possibility of model interpretation and analysis.

- We provide an extensive evaluation of the model against both classical and state-of-the-art methods on a set of 33 regression UCI open access datasets, the Feynman symbolic regression benchmark with 120 datasets, the Friedman-1 dataset, and the publicly available multi-target dataset from the gas and petrochemical processing.
- We provide detailed analyses to understand model's performance: ablation study, the impact of data properties with different numbers of unimportant features, noise levels, and number of samples.

Method

To describe the proposed regression model for the problem (1), let us first define a Taylor map as the transformation $\mathcal{M} : \mathbf{Z}_t \to \mathbf{Z}_{t+1}$ in the form of

$$\mathbf{Z}_{t+1} = \mathcal{M}(\mathbf{Z}_t) = W_0 + W_1\,\mathbf{Z}_t + \ldots + W_k\,\mathbf{Z}_t^{[k]}, \quad (3)$$

where $\mathbf{Z}_t, \mathbf{Z}_{t+1} \in \mathbb{R}^{n+m}$, matrices $W_0, W_1, \ldots, W_k$ are trainable weights, and $\mathbf{Z}^{[k]}$ means k-th Kronecker power of vector $\mathbf{Z}$. The transformation (3) can be referred to Taylor maps and models (Berz 1997), exponential machines (Novikov, Trofimov, and Oseledets 2017), tensor decomposition (Dolgov 2019), and others. In fact, map $\mathcal{M}$ is just a multivariate polynomial regression of k-th order.

Proposed Model

Let's now extend feature vector $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ in the regression problem (1) with additional m dimensions filled with zeros and consistently apply the map (3) p times:

$$\begin{aligned} &\mathbf{Z}_0 = (\mathbf{x}_1, \ldots, \mathbf{x}_n, 0, \ldots, 0), \\ &\mathbf{Z}_1 = W_0 + W_1\,\mathbf{Z}_0 + \ldots + W_k\,\mathbf{Z}_0^{[k]}, \\ &\ldots \\ &\mathbf{Z}_p = W_0 + W_1\,\mathbf{Z}_{p-1} + \ldots + W_k\,\mathbf{Z}_{p-1}^{[k]}. \end{aligned} \quad (4)$$

This procedure implements Taylor mapping that propagates initial vector $\mathbf{Z}_0$ along new discrete dimension $t = \{1, 2, \ldots p\}$ with hidden states $\mathbf{Z}_t = (x_{1,t}, \ldots, x_{n,t},\, y_{1,t}, \ldots, y_{m,t})$.

For the final state $\mathbf{Z}_p$ this yields a polynomial of order k^p with respect to the components of $\mathbf{Z}_0$ which is, however, factorized by a Taylor map $\mathcal{M}$ of order k:

$$\mathbf{Z}_p = \mathcal{M} \circ \ldots \circ \mathcal{M} \circ \mathbf{Z}_0 = V_0 + V_1\,\mathbf{Z}_0 + \ldots + V_{k^p}\,\mathbf{Z}_0^{[k^p]},$$

where weight matrices $V_q = V_q(W_0, W_1, \ldots, W_k)$ for $q = \{0, 1, \ldots k^p\}$. For defining a loss function, one should consider only the last m components of the vector $\mathbf{Z}_p = (x_{1,p}, \ldots, x_{n,p},\, y_{1,p}, \ldots, y_{m,p})$ as the predictions:

$$Loss = \rho((\mathbf{y}_1, \ldots, \mathbf{y}_m), (y_{1,p}, \ldots, y_{m,p})), \quad (5)$$

where ρ is an error between true targets $(\mathbf{y}_1, \ldots, \mathbf{y}_m)$ and predicted values $(y_{1,p}, \ldots, y_{m,p})$.

Fig. 1 presents this algorithm as a Taylor map PNN (TMPNN) with layers of shared weights that propagate the extended vector of features $\mathbf{Z}_0$. Namely, after the first layer, all variables in the feature vector are modified, and the variables on the m introduced dimensions become non-zero in general. The mapping (3) $\mathbf{Z}_t \to \mathbf{Z}_{t+1}$ continues until the p-th layer, where the last m variables are used for the output.

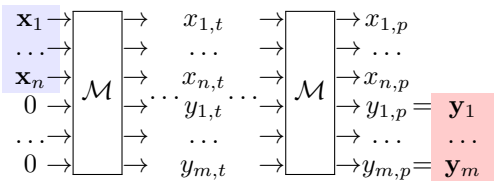

Figure 1: High-order regression $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n) \to (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_m)$ factorized by Taylor map $\mathcal{M}$ and implemented as PNN with p layers of shared weights.

If the number of layers $p = 1$, the proposed architecture is equivalent to regular polynomial regression (2) of order k. If the order of the Taylor map $k = 1$, then the proposed model is equivalent to linear regression. The proposed architecture defines a high-order polynomial regression of order k^p for other cases. Since the weights of $\mathcal{M}$ are shared, the number of free parameters in the model equals to $(n + m)\sum_{l=0}^{k} C_{n+m-1+l,n+m-1}$, where a combination $C_{a+b,a}$ defines the number of the monomials of degree b with $a + 1$ variables. This model can result in extremely high-order polynomials with significantly fewer free parameters than regular polynomial regression.

The architectures of polynomial transformations with shared weights have been previously used in various research topics but were not formulated for general-purpose regression. For example, (Dragt, Gjaja, and Rangarajan 1991) used Taylor maps to factorize dynamics in Hamiltonian systems. (Ivanov and Agapov 2020) applied the same concept for data-driven control of X-ray sources. (Chrysos et al. 2020) described a similar computational graph with polynomial shared layers for applications in computer vision. The current paper considers the algorithm (4) as a general-purpose regression model for multi-target problems.

Model Hyperparameters

The proposed model has two fundamental hyperparameters: the order k of the nonlinearities in the map (3) and the number p of iterations in (4). These parameters define the order k^p of the polynomials that represent the relationship between the input variables and targets.

We speculate that in addition to defining the order k^p of the polynomial, the number of steps p also serves as a regularization. More iterations in (4) result in larger derivatives for the high-order terms in the loss. While we can claim empirical evidence for this behavior, the theoretical properties should be examined in additional research. Moreover, one can introduce classical regularization, such as L1, L2, or dropout, and incorporate their respective hyperparameters.

Finally, the initial values $(y_{1,0}, \ldots, y_{m,0})$ in $\mathbf{Z}_0$ can be considered as free parameters. Instead of extending the input vector $\mathbf{X}$ with zeros, it is possible to consider any other values, resulting in a new set of m hyperparameters.

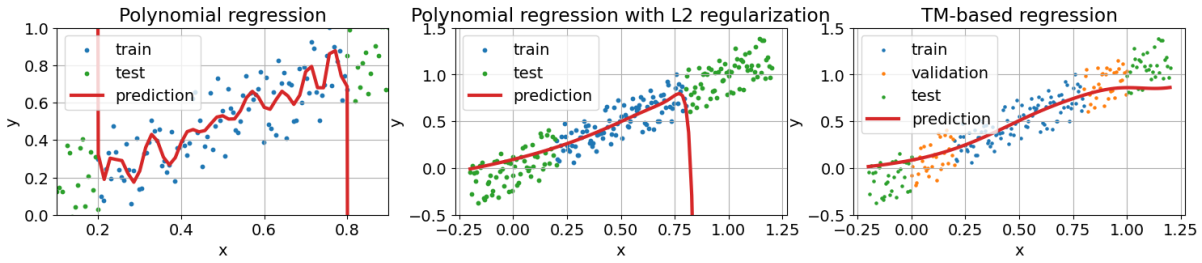

Figure 2: Comparison of regular 41st order polynomial regression (left plot), regularized one (central plot), and proposed regression of 125th order (right plots) for the fitting of noisy linear data. Each of the models has 41 weight coefficients.

Implementation

A naive implementation of the proposed regression model can be done in Python using a simple **for** loop. An example of the second-order mapping (3) is presented in Listing 1.

More advanced implementation using Keras and TensorFlow is provided in https://github.com/andiva/tmpnn. It includes a layer that implements a Taylor map (3) of arbitrary order of nonlinearities and the construction of the computational graph (4). We use the Adamax optimizer (Kingma and Ba 2015) with default parameters based on our experiments. The paper does not cover the choice of the optimal optimization method for the proposed model.

Listing 1: Proposed regression for $k = 2$ and $p = 10$

```
def predict(X, W0, W1, W2, num_targets):
    num = X.shape[0] # number of samples
    Y0 = np.zeros((num, num_targets))
    # extend X with zeros:
    Z = np.hstack((X, Y0))
    for _ in range(10): # p=10
        # 2nd kronecker power:
        Z2 = (Z[:,:,None]*Z[:,None,:])
        Z2 = Z2.reshape(num, -1)
        Z = W0 + np.dot(Z, W1) + np.dot(
            Z2, W2)
    # last num_targets values:
    return Z[:, -num_targets:]
```

Example

To demonstrate the main difference of (4) in comparison to the regular polynomial regression (2), the linear model $\mathbf{y} = \mathbf{x} + \epsilon$ with a single variable $\mathbf{x}$, a single target $\mathbf{y}$, and random noise $\epsilon \sim U(-0.25, 0.25)$ is utilized.

For the regular polynomial regression (2), we use a polynomial of 41-st order just for example

$$\hat{\mathbf{y}}_{pred} = c_0 + c_1\mathbf{x} + \ldots + c_{41}\mathbf{x}^{41}, \tag{6}$$

and try using both non-regularized and L2-regularized mean squared error (MSE). Fig. 2, central plot, reports a regularization parameter set at 3. We explored the range from 0.001 to 1000 but observed no significant differences in poor prediction within out-of-distribution ranges.

For the proposed model (4), we use fifth order map (3) and three steps in (4). This results in 125-th order polynomial $\mathbf{y}_{pred} = y_3$ with $x_0 = \mathbf{x}$ and $y_0 = 0$ incorporating the same number of free parameters:

$$\begin{pmatrix} x_{t+1} \\ y_{t+1} \end{pmatrix} = \begin{pmatrix} w_0 \\ w_1 \end{pmatrix} + \begin{pmatrix} w_2 & w_3 \\ w_4 & w_5 \end{pmatrix} \begin{pmatrix} x_t \\ y_t \end{pmatrix} + \begin{pmatrix} w_6 & w_7 & w_8 \\ w_9 & w_{10} & w_{11} \end{pmatrix} \begin{pmatrix} x_t^2 \\ x_t y_t \\ y_t^2 \end{pmatrix} + \ldots + \begin{pmatrix} w_{30} & w_{31} & \ldots & w_{35} \\ w_{36} & w_{37} & \ldots & w_{41} \end{pmatrix} \begin{pmatrix} x_t^5 \\ x_t^4 y_t \\ \ldots \\ y_t^5 \end{pmatrix}. \tag{7}$$

As expected, polynomial regression (6) of 41-st order leads to overfitting, resulting in a curve with many kinks attempting to predict the noise. L2 regularization shrinks all coefficients c_i to zero but still does not help with poor extrapolation (Fig. 2, left and central plots).

The proposed model, instead of fitting one-dimensional curve, attempts to fit a surface in the three-dimensional space (x_t, y_t, t). Starting with curve $(\mathbf{x}, 0, 0)$, the model propagates it to the $(x_3, \mathbf{y}_{pred}, 3)$. The visualization of this approach is provided in Fig. 3. The fitted model (7) without any extra regularization has coefficients w_i ranging from -0.075 to 0.11. As a result, the final polynomial of 125-th order is smoother, providing a better extrapolation for a wider region of unseen data (see Fig. 2, right plot).

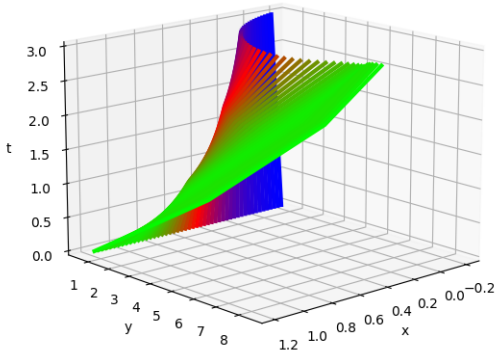

Figure 3: Surface fitted in the space (x_t, y_t, t) with the proposed high-order regression (4) for linear data.

Interpretation and Approximation

In this section, we demonstrate that the TMPNN can be considered a numerical approximation of the general solution for a system of ODEs that approximates problem (1). Note that although the TMPNN is theoretically grounded with ODEs, the model, as a PNN with shared weights, eliminates the need for numerical solvers for ODEs during training.

Interpretation

The Taylor map (3) can be written in the form

$$\begin{aligned}\mathbf{Z}_{t+1} = \mathbf{Z}_t + \Delta\tau(pW_0 + p(W_1 - I)\mathbf{Z}_t \\ + pW_2\,\mathbf{Z}_t^{[2]} + \ldots + pW_k\,\mathbf{Z}_t^{[k]}),\end{aligned} \tag{8}$$

with $\Delta\tau = 1/p$ and I for identity matrix. Comparing (8) with the simplest Euler method for solving ODEs with a right-hand side F:

$$\begin{aligned}\frac{d}{d\tau}\mathbf{Z}(\tau) = F\left(\mathbf{Z}(\tau)\right), \\ \mathbf{Z}(\tau + \Delta\tau) = \mathbf{Z}(\tau) + \Delta\tau F\left(\mathbf{Z}(\tau)\right),\end{aligned} \tag{9}$$

one can conclude that Taylor map (3) represents the difference equations for a discrete state $\mathbf{Z}_t$ for the evolution of the state vector $\mathbf{Z}(\tau) = (x_1(\tau), \ldots, x_n(\tau), y_1(\tau), \ldots, y_m(\tau))$ in continuous time $\tau \in [0, 1]$ with time step $\Delta\tau = 1/p$:

$$\frac{d}{d\tau}\begin{pmatrix} x_1 \\ \ldots \\ x_n \\ y_1 \\ \ldots \\ y_m \end{pmatrix} = A_0 + A_1\begin{pmatrix} x_1 \\ \ldots \\ x_n \\ y_1 \\ \ldots \\ y_m \end{pmatrix} + A_2\begin{pmatrix} x_1 \\ \ldots \\ x_n \\ y_1 \\ \ldots \\ y_m \end{pmatrix}^{[2]} + \ldots, \tag{10}$$

where $A_q = pW_q$ for $q \neq 1$ and $A_1 = p(W_1 - I)$.

Furthermore, the proposed model (4) corresponds to the boundary value problem for (10) with the initial conditions

$$\begin{aligned}x_1(0) = \mathbf{x}_1, x_2(0) = \mathbf{x}_2, \ldots, x_n(0) = \mathbf{x}_n, \\ y_1(0) = 0, \ldots, y_m(0) = 0\end{aligned} \tag{11}$$

and the boundary conditions

$$y_1(\tau = 1) = \mathbf{y}_1, \ldots, y_m(\tau = 1) = \mathbf{y}_m. \tag{12}$$

Otherwise stated, the regression model (4) with trainable weights W_q is equivalent to the boundary value problem (11) and (12) for the system of ODEs (10) with trainable weights A_q. The particular solution $\mathbf{Z}(\tau)$ of the system (10) for the initial conditions (11) at time $\tau = t/p$ corresponds to the discrete state $\mathbf{Z}_t$ after the t-th step in the (4).

The connection between the proposed model (4) and ODEs highlights that both features and targets, as well as all targets together, are coupled. It provides a natural way to construct a multi-target regression without building multiple single-output models. Moreover, since the trainable weights W_q correspond to A_q in the system (10), this approach allows us to reconstruct the system of ODEs that approximates the training dataset without the need to solve differential equations numerically. Only the weights in (4) are tuned with the data during training.

Approximation Capabilities

Technical assumptions: The feature space $\mathcal{X} \subset \mathbb{R}^n$ forms a compact Hausdorff space (separable metric space). The feature distribution $\mathbb{P}_{\mathbf{X}}$ is a Borel probability measure.

Besicovitch assumption: The regression function $\eta : \mathcal{X} \to \mathbb{R}^m$ satisfies $lim_{r\to+0}\mathbb{E}\left[\mathbf{Y}|\mathbf{X} \in B_{\mathbf{x},r}\right] = \eta(\mathbf{x})$ for $\mathbf{x}$ almost everywhere w.r.t. $\mathbb{P}_{\mathbf{X}}$, with $B_{\mathbf{x},r}$ standing for a ball with radius r around $\mathbf{x}$.

Lemma: For any function $\eta \in \mathcal{L}_p$ (Lebesgue spaces) and tolerance $\forall\varepsilon > 0$ a continuously differentiable function $f \in C^1$ exists such that a norm in $\mathcal{L}_p$ $||\eta(\mathbf{x}) - f(\mathbf{x})||_p \leq \varepsilon$ under technical assumptions. See prof in (Folland 2007).

Theorem: *Under the given assumptions and lemma, for any function $\eta \in \mathcal{L}_p$ and tolerance $\forall\varepsilon > 0$ parameters k and p in the model* (4) *exist such that $||y(p) - f(\mathbf{x})||_p \leq \varepsilon$.*

Proof. To prove this theorem, without loss of generality, we consider the example of a scalar function $\mathbf{y} = \eta(\mathbf{x})$ of one variable. Let's first approximate it with a given tolerance $\varepsilon/2$ with continuously differentiable function $f(\mathbf{x})$. By introducing two variables $x(\tau)$ and $y(\tau)$, one can derive a system of ODEs that implements mapping $y(0) = 0 \to y(1) = f(\mathbf{x})$. For this one can suppose $y = \xi(\tau, x)f(x)$ with conditions $\xi(0, x) \equiv 0, \xi(1, x) \equiv 1$, and write the differential of ξ:

$$\frac{\partial\xi}{\partial\tau}d\tau + \frac{\partial\xi}{\partial\mathbf{x}}d\mathbf{x} = (fdy - y\frac{\partial f}{\partial\mathbf{x}}dx)/f^2.$$

Using this relation, one can write a system

$$\begin{aligned}f(x)\frac{dy}{d\tau} = \frac{\partial\xi}{\partial\tau}f^2(x) + \Theta(x, y, \tau), \\ \left(y\frac{\partial f}{\partial\mathbf{x}} + f^2(x)\frac{\partial\xi}{\partial\mathbf{x}}\right)\frac{dx}{d\tau} = \Theta(x, y, \tau),\end{aligned} \tag{13}$$

where $\Theta(x, y, \tau)$ is chosen arbitrary. Equations (13) can then be written in the form of ODEs by choosing $\Theta(x, y, \tau) = \nu(x, y, \tau)\left(f(x)y\frac{\partial f}{\partial\mathbf{x}} + f^2(x)\frac{\partial\xi}{\partial\mathbf{x}}\right)$:

$$\begin{aligned}\frac{dx}{d\tau} = \nu f(x), \\ \frac{dy}{d\tau} = \frac{\partial\xi}{\partial\tau}f(x) + \nu f^2(x)\frac{\partial\xi}{\partial x} + \nu y\frac{\partial f(x)}{\partial x},\end{aligned} \tag{14}$$

which together with the initial conditions $x(0) = \mathbf{x}, y(0) = 0$ leads to the desired solution $y(1) = f(\mathbf{x})$.

The system (14) represents one of many possible ODEs that the model (4) implicitly learns in its discrete step-by-step representation. Moreover, in the case of $\nu = 0, \xi = \tau$, the system (14) corresponds to the regular polynomial regression (2). Indeed, in this case $\frac{dx}{d\tau} = 0$ and $\frac{dy}{d\tau} = f(x)$. If the function $f(\mathbf{x})$ is a polynomial of k-th order, the Taylor map $y_{t+1} = y_t + f(x(0))/p$ with the initial condition $x(0) = \mathbf{x}$, $y_0 = y(0) = 0$ exactly produces regular polynomial regression $\mathbf{y} = y_p = f(\mathbf{x})$. For the non-polynomial functions $f(\mathbf{x})$, the systems (14) still corresponds to the output $\mathbf{y} = y(1) = f(\mathbf{x})$, but the Taylor mapping (4) approximates this solution under technical assumptions with an arbitrary accuracy $\varepsilon/2$ depending on the order k and the number of steps p due to the Stone-Weierstrass theorem (Weierstrass 1885; Stone 1948). Thus, the Taylor mapping also approximates a given regression function η with ε accuracy.

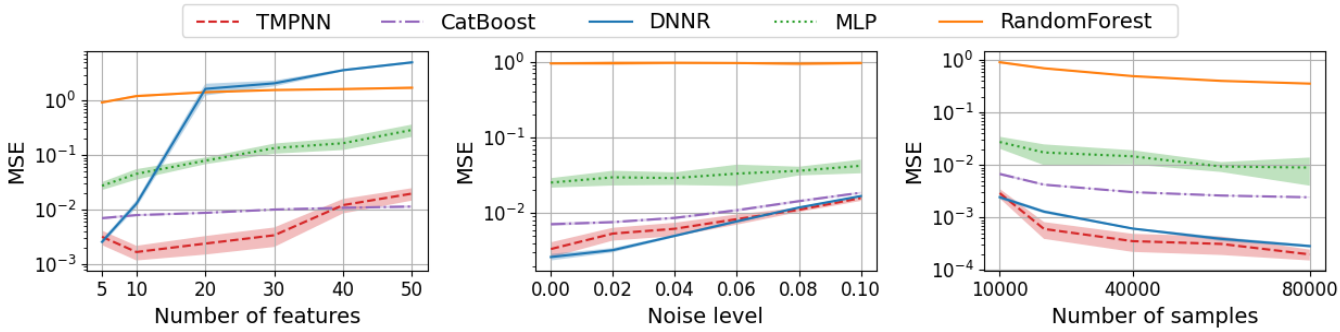

Figure 4: The effect of different sampling conditions on model performance for the Friedman-1 dataset.

Remark. The exact forms of ν and ξ are chosen for the simplicity of the proof. In the general case (14), they are arbitrary, providing more freedom for the optimization process. The resulting convergence and accuracy of (4) with respect to parameters k and p can be theoretically estimated (Andrianov 2012; Iben and Wagner 2021).

Increasing the Order of Polynomial Regression

One can increase order k or add steps p in (4) to generate a higher order polynomial regression. Rising the order increases the complexity of the model while increasing steps preserves the same number of trainable parameters but makes TMPNN deeper, which can affect optimization performance without proper weights initialization.

Previous section provides an approach for the weights initialization. Given the model (4) with parameters k, p, and trained weights W_i, one can write the system (10) and generate a new model (4) by integrating (10) with the increased number of steps $\bar{p} > p$. This results in the new weights

$$\begin{aligned} \bar{W}_q &= pW_q/\bar{p}, \quad q \neq 1, \\ \bar{W}_1 &= pW_1/\bar{p} + (\bar{p} - p)I/\bar{p} \end{aligned} \tag{15}$$

for the TMPNN with more steps and, as a result, higher polynomial of order $k^{\bar{p}}$. This approach transforms the lower-order model k, p to a higher-order $k, \bar{p}$, providing a proper weight initialization.

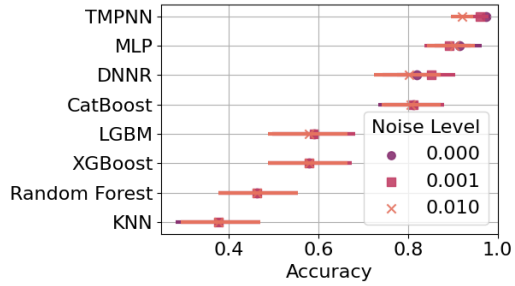

Figure 5: Accuracy shows the percentage of solutions with R2 $>$ 0.999 on the Feynman Regression dataset under three noise levels. The bars denote 95% confidence intervals.

Experiments

The experiments setup and hyperparameters for the considered models can be found in supplementary code.

Interpolation

Feynman Benchmark Feynman Symbolic Regression Database consists of 120 datasets sampled from physics equations (Udrescu and Tegmark 2020). These equations are continuous differentiable functions. We increased the difficulty of the datasets by adding Gaussian noise with three standard deviations (0, 0.001, 0.01). The evaluation was executed with ten different random splits (75% for training, 25% for testing and metric reporting).

We define the accuracy as the percentage of datasets solved with the coefficient of determination R2 $>$ 0.999 and report it in Fig. 5. The accuracy of the classical and state-of-the-art models was taken from (Nader, Sixt, and Landgraf 2022), where optimal hyperparameters search was performed for the same datasets. We do not use optimal hyperparameter search for the proposed model and suppose $k = 3, p = 5$ as default. The proposed model is the best-performing one with the smallest confidence intervals independently of the noise level.

UCI datasets For the 32 UCI regression datasets with less than 50 features[1], we compare TMPNN ($k = 2, p = 7$), CatBoost, and DNNR with default parameters. For the evaluation of each model, we consider only datasets with $R2 > 0.5$. CatBoost and TMPNN provide the same average $R2 = 0.88 \pm 0.14$ on 26 datasets. DNNR results in average $R2 = 0.87 \pm 0.12$ but fits only 18 datasets.

Effect of Noise, Number of Samples and Features

This section investigates how the noise, the number of samples, and presence of unimportant features affect the model's performance. Since such an analysis requires a controlled environment, we used the classical Friedman-1 dataset that allows varying sampling conditions. Friedman-1 dataset is generated for five uniformly distributed in $[0, 1]$ features. The noise is sampled from a standard normal distribution. Additional unimportant features are added by sampling from the continuous uniform distribution $U(0, 1)$.

[1]https://github.com/treforevans/uci_datasets to access datasets

	Sarcos	CO^2Emission	California	Airfoil	Concrete	NOxEmission	Pendulum
CatBoost	1.71±0.06	**1.02±0.13**	**0.19±0.01**	**1.34±0.30**	**15.34±7.66**	14.73±1.12	4.55±2.58
DNNR	**0.80±0.05**	1.28±0.13	0.24±0.02	4.56±0.86	35.98±9.01	18.39±1.78	2.83±1.64
TabNet	1.56±0.25	1.12±0.21	0.39±0.03	**1.36±0.39**	18.50±6.27	**11.17±1.25**	2.40±1.10
XGBoost	2.12±0.06	1.14±0.14	0.21±0.01	1.62±0.33	**16.90±7.92**	16.21±0.89	5.09±2.68
MLP	1.17±0.12	1.08±0.21	0.30±0.02	6.18±1.57	22.72±7.57	14.78±1.57	**1.80±0.98**
KNN	2.47±0.09	1.10±0.18	0.40±0.01	8.26±1.06	70.73±13.63	18.54±1.10	4.03±2.03
TMPNN	1.40±0.05	1.27±0.26	0.38±0.01	3.15±1.06	26.46±4.47	17.30±4.90	2.45±1.10

Table 1: The MSE on various UCI datasets averaged over random 10-fold cross-validation. The standard deviations are given after the ± signs. TMPNN provides moderate results if the assumptions on data are violated.

Besides TMPNN, we also evaluated CatBoost, DNNR, MLP, and Random Forest. We use the same train-test splitting as for the Feynman benchmark. The hyperparameters for each method are fitted on the default dataset (10,000 samples, five features, without noise) and are fixed for the remaining analysis. Fig. 4 reports the effect of each condition. TMPNN performs the best for the low number of added unimportant features but is beaten by CatBoost slightly for higher numbers. TMPNN is the only model that demonstrates error reduction when adding a small number of unimportant features. This behavior refers to the extra latent dimensions and is discussed briefly in conclusion. For the noise level, CatBoost, DNNR, and TMPNN perform similarly. For the number of samples, TMPNN is the best model outperforming even DNNR, which is based on nearest neighbors.

Ablation Study

In this section, we discuss TMPNN performance under various design alternatives, such as initialization of weights, number of layers p, nonlinear orders k, as well as presence of categorical features. We base this analysis on the specific set of UCI datasets, Airfoil dataset and 10,000 samples from Friedman-1 dataset (Friedman 1991). We use ten random splits similar to Feynman benchmarking for each of the settings.

Table 1 demonstrates the performance of the proposed model (TMPNN) for several UCI and Sarcos open access datasets. For evaluation, we used several state-of-the-art and classical models. CatBoost is the best-performing method. TMPNN demonstrates comparable results against other models serving as the second and third-best model for the Pendulum and Sarcos datasets. Since these datasets contain categorical features and, moreover, some of them violate the assumption that the datasets are defined by a physical system, TMPNN demonstrates just average results.

Table 2 reports performance of the TMPNN under various model design alternatives. Initializations differed from identity mapping (3) lead to worse results for both datasets. Increasing the number of layers starting at $p = 5$ results in a decrease in performance. We suppose that number of layers affects the optimizer resulting in convergence issues because we did not use relation (15) for weights initialization.

	(k, p)	Airfoil	Friedman-1
$W_i = 0$ $i \neq 1$ $W_1 = I$	(2, 5)	0.87 ± 0.01	0.99 ± 0.00
	(3, 5)	0.92 ± 0.01	0.99 ± 0.00
	(3, 10)	0.89 ± 0.05	0.99 ± 0.00
	(3, 20)	0.87 ± 0.11	0.99 ± 0.00
	(4, 5)	0.80 ± 0.12	0.99 ± 0.00
$W_i = 0$	(2, 5)	0.66 ± 0.02	0.99 ± 0.00
	(3, 5)	-1.07 ± 2.44	0.99 ± 0.00
	(4, 5)	-3.86 ± 4.29	–
$W_i = \epsilon_i$, $i \neq 1$ $W_1 = I + \epsilon_1$	(2, 5)	0.86 ± 0.02	0.99 ± 0.00
	(3, 5)	0.91 ± 0.1	0.99 ± 0.00
	(4, 5)	0.74 ± 0.24	–

Table 2: R2 score for variations of TMPNN. Sign "–" stands for the failure to converge. Noise $\epsilon_j \subset \mathcal{N}(0, 0.0001)$.

Extrapolation

UCI Yacht Hydrodynamics dataset The UCI Yacht Hydrodynamics dataset consists of 308 experiments connecting six features (lcg, cp, l/d, b/d, l/b, fn) with a target (rr) (Gerritsma et al. 2013). We compare the extrapolation ability of models using cross-validation for hyperparameters choice.

We use several train-test splits for extrapolation when samples exceeding 75% quantile by each feature and target are chosen for test sets. Table 3 presents R2 scores. The first row refers to the rule for test dataset generation and its size in percentage. CatBoost, DNNR and MLP are unsuitable for out-of-distribution prediction depending on feature fn and target rr. In these cases, the test set contains completely different values for the target. Regular polynomial regression and TabNet perform better but are still unstable. TMPNN provides the best scores.

	cp>0.5 (22%)	b/d>4.1 (22%)	fn>0.4 (21%)	rr>12 (25%)
MLP	0.94	0.88	-0.53	-0.49
CatBoost	0.98	0.97	-3.27	-2.40
DNNR	**0.99**	**0.99**	-0.46	-0.56
Polyn. Regr.	0.63	-0.61	0.89	0.83
TabNet	**0.99**	**0.99**	0.71	-0.01
TMPNN	**0.99**	**0.99**	**0.92**	**0.90**

Table 3: R2 scores on various extrapolation tests for different models on UCI Yacht Hydrodynamics dataset.

	MAPE, %			size,
	train	public test	private test	MB
DNNR	1.04	18.67	19.62	296
CatBoost	1.52	9.97	10.09	2.8
TabNet	0.85	5.36	5.34	1.7
Ridge + SVR	1.32	1.72	1.72	20.5
TMPNN	**0.39**	**1.02**	**1.05**	**0.2**

Table 4: Various models submitted to Sibur2023 Challenge. MAPE is provided for public train and hidden test datasets. Last column stands for the serialized model size.

Gas Processing Dataset

The proposed model won the first prize in the Sibur2023 Online Challenge[2] hosted by SIBUR company, a leading emerging markets petrochemical group. The task aimed to predict the volumes of two products based on raw material type and production process parameters. The key requirement was out-of-distribution prediction which challenges the design of experiments in data-scarce scenarios.

The training dataset consists of 153,417 samples with 25 features and two targets. The test dataset, undisclosed by the organizers, contains 480,656 samples with feature values not presented in the public training dataset. CatBoost regressor, the baseline model, achieves a MAPE of 1.5% for train and 10% for test datasets caused by out-of-distribution samples.

Table 4 presents the performance of several submitted solutions. For the TMPNN, we used $k = 2, p = 5$ with all features and without feature engineering, selection, and regularization. Iinstead of using zeros in the extended vector $\mathbf{Z}_0$, we process $y_1(0), y_2(0)$ as hyperparameters.

TMPNN achieved outstanding performance by fitting a 32nd-order polynomial over 25 features. The model prevented overfitting without the need for additional regularization. This practical example highlights the specific applicability of the presented approach for control systems in manufacturing, especially considering small model size in comparison to state-of-the art models.

Further Development

Theoretical results from fields of PNNs and ODEs can be utilized to estimate the error of the proposed model and explore the convergence of the training w.r.t. number of epochs, hyperparameters, and presence of noise in the data.

The extrapolation ability of the proposed model can be explored by drawing upon the theory of ODEs. As the model implicitly learns the general solution of a system of ODEs, the out-of-distribution inference can be viewed as new initial conditions for already learned general solution.

Due to the Picard-Lindelöf theorem for ODEs, the proposed model can be utilized for classification problems constraining the trajectories $\mathbf{Z}_t$ to converge to the target states $\mathbf{Z}_c \in \mathbb{R}^{n+m}$ for each class c. While the applicability of PNNs for classification has been discussed in (Chrysos et al. 2021), it should be extended to the proposed model.

[2]https://platform.aitoday.ru/event/9

Regarding feature engineering, there are two possibilities for improving the model. Firstly, instead of initializing the state $\mathbf{Z}_0 \in \mathbb{R}^{n+m}$ in (4) with the last m zeros, one can use learnable parameters, custom functions, or predictions from other models. Secondly, one can extend the state vector $\mathbf{Z}_0$ by appending l latent units resulting in $\mathbf{Z}_0 \to \mathbf{Z}_E \in \mathbb{R}^{n+m+l}$. This increases the expressivity of the model and has been successfully applied in regression tasks (Chen et al. 2018; Green and Rindler 2019).

While TMPNN provides a way to build a polynomial regression of order k^p, the approach (15) allows increasing the order of polynomial even more. The iterative weights initialization helps to increase the number of layers p, resulting in a higher polynomial if necessary.

The possible limitations of the TMPNN need to be considered in more detail. These include the influence on data normalization, presence of categorical and ordinal features, and the differentiability of the functional relationship between targets and features. Investigating these aspects is crucial for effectively applying the model to diverse datasets.

Conclusion

As mentioned above, certain theoretical aspects regarding the proposed model are not addressed within the paper. The main goal of the current work is to present a novel approach for constructing extremely high-order polynomials to solve regression problems and empirically demonstrate its validity. There are two main contributions in the paper. Firstly, we introduced a high-order polynomial model designed to accommodate multi-target regression. The key for understanding of TMPNN is replacing regular 'curve fitting' $\mathbf{x} \to \mathbf{y}$ with 'surface fitting' in the artificial timeline $(x, y, t) : (\mathbf{x}, 0, 0) \to (x_p, \mathbf{y}, p)$. Secondly, we presented the interpretation and approximation capabilities of the model by establishing its connection with ODEs.

The assumption that the higher-order monomials of the learned polynomial are in some form related to the lower order ones is grounded in the well-studied numerical analysis of step-by-step integration solvers. If there is an evidence that the dataset is generated by a dynamical system, the proposed model is the primary choice. Otherwise, it is just another model to test among other ML models.

From a theoretical standpoint, the differential viewpoint of the polynomial makes sense when p approaches infinity, corresponding to the infinitesimal step of $1/p$. On the other hand, similar to numerical solvers, TMPNN practically treats p as a finite hyperparameter. As shown in the experiments, TMPNN can effectively learn imperfections caused by its finite iterative scheme even with relatively low values of p. In our experiments, up to ten steps were used.

With the example of 155 datasets and different sampling conditions such as noise level, number of samples, and presence of unimportant features, we demonstrated that the proposed model is well-suited for regression tasks and outperforms state-of-the-art methods on specific problems. Moreover, connection of the TMPNN with ODEs makes it naturally applicable to various domains such as physical, chemical, or biological systems. At the same time, TMPNN does not require numerical solvers for ODEs during training.

Acknowledgments

The authors thank the reviewers for helpful comments and clearer formulations of statements. We are also grateful to Prof. Dr. Uwe Iben for the discussion of the proposed model and its applicability.

References

Andrianov, S. 2012. The convergence and accuracy of the matrix formalism approximation. *Proc. of the 11th Intern. Comp. Accelerator Physics. Conference*, 93–95.

Arik, S. O.; and Pfister, T. 2021. Tabnet: Attentive interpretable tabular learning. *Proc. of the AAAI Conf. on Artificial Intelligence*, 35(8): 6679–6687.

Berz, M. 1997. From Taylor series to Taylor models. *AIP Conference Proceedings*, 405(1): 1–23.

Blondel, M.; Ishihata, M.; Fujino, A.; and Ueda, N. 2016. Polynomial Networks and FM: New Insights and Efficient Training Algorithms. *Proceedings of the 33rd Intern. Conf. on Machine Learning*.

Borchani, H.; Varando, G.; Bielza, C.; and Larrañaga, P. 2015. A survey on multi-output regression. *WIREs Data Mining and Knowledge Discovery*, 5(5): 216–233.

Chen, T. Q.; Rubanova, Y.; Bettencourt, J.; and Duvenaud, D. K. 2018. Neural ordinary differential equations. In *Advances in Neural Information Proc. Systems*, 6571–6583.

Chrysos, G.; Moschoglou, S.; Bouritsas, G.; Panagakis, Y.; Deng, J.; and Zafeiriou, S. 2020. $\Pi-$nets: Deep Polynomial Neural Networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 7325–7335.

Chrysos, G. G.; Georgopoulos, M.; Deng, J.; and Panagakis, Y. 2021. Polynomial Networks in Deep Classifiers. https://arxiv.org/abs/2104.07916v1. Accessed: 2023-12-31.

Davierwalla, D. 1977. *REGSTEP - stepwise multivariate polynomial regression with singular extensions*, volume 7:48905. Würenlingen: EIR. EIR-Bericht.

Dette, H. 1995. Optimal Designs for Identifying the Degree of a Polynomial. *Ann. Statist.*, 23(4): 1248–1266.

Dolgov, S. 2019. A tensor decomposition algorithm for large ODEs with conservation laws. *Computational Methods in Applied Mathematics*, 19(1): 23–38.

Dorogush, A.; Ershov, V.; and Gulin, A. 2018. CatBoost: gradient boosting with categorical features support. *https://arxiv.org/abs/1706.09516*.

Dragt, A.; Gjaja, I.; and Rangarajan, G. 1991. Kick factorization of symplectic maps. In *Conference Record of the 1991 IEEE Particle Accelerator Conference*, 1621–1623.

Fan, J.; Gasser, T.; Gijbels, I.; Brockmann, M.; and Engel, J. 1997. Local Polynomial Regression: Optimal Kernels and Asymptotic Minimax Efficiency. *Annals of the Institute of Statistical Mathematics*, 49: 79–99.

Folland, G. 2007. *Real Analysis: Modern Techniques and Their Applications*. Wiley, 2nd edition.

Freudenthaler, C.; Schmidt-Thieme, L.; and Rendle, S. 2011. Factorization Machines Factorized Polynomial Regression Models. *IEEE International Conference on Data Mining, Sydney*.

Friedman, J. H. 1991. Multivariate adaptive regression splines. *The Annals of Statistics*, 19(1): 1–67.

Fronk, C.; and Petzold, L. 2023. Interpretable polynomial neural ordinary differential equations. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 33(4): 043101.

Gerritsma, J.; Onnink, R.; ; and Versluis, A. 2013. Yacht Hydrodynamics. https://doi.org/10.24432/C5XG7R. Accessed: 2023-12-31.

Green, D.; and Rindler, F. 2019. Model inference for ODEs by parametric polynomial kernel regression. *3rd ECCOMAS Thematic Conference on Uncertainty Quantification in Computational Sciences and Engineering*.

Hofwing, M.; Strömberg, N.; and Tapankov, M. 2011. Optimal Polynomial Regression Models by using a Genetic Algorithm. *Civil-Comp Proceedings*, 97.

Iben, U.; and Wagner, C. 2021. Taylor mapping method for solving and learning of dynamic processes. *Inverse Problems in Science and Engineering*, 29:13: 3190–3213.

Ivanov, A.; and Agapov, I. 2020. Physics-based deep neural networks for beam dynamics in charged particle accelerators. *Phys. Rev. Accel. Beams*, 23: 074601.

Ivanov, A.; Golovkina, A.; and Iben, U. 2020. Polynomial Neural Networks and Taylor Maps for Dynamical Systems Simulation and Learning. *24th Europ. Conf. on AI 2020, Spain*.

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In Bengio, Y.; and LeCun, Y., eds., *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Lewis, M. 2007. Stepwise versus Hierarchical Regression: Pros and Cons. https://files.eric.ed.gov/fulltext/ED534385.pdf. Accessed: 2023-12-31.

López, V.; Huerta, R.; and Dorronsoro, J. R. 1993. Recurrent and Feedforward Polynomial Modeling of Coupled Time Series. *Neural Comput.*, 5(5): 795–811.

Nader, Y.; Sixt, L.; and Landgraf, T. 2022. DNNR: Differential Nearest Neighbors Regression. *Proc. of the 39 th Intern. Conf. on Machine Learning*.

Novikov, A.; Trofimov, M.; and Oseledets, I. 2017. Exponential Machines. https://arxiv.org/abs/1605.03795. Accessed: 2023-12-31.

Oh, S.-K.; Pedrycz, W.; and Park, B.-J. 2003. Polynomial neural networks architecture: Analysis and design. *Computers & Electrical Engineering*, 29: 703–725.

Pakdemirli, M. 2016. A New Perturbation Approach to Optimal Polynomial Regression. *Mathematical and Computational Applications*, 21: 1.

Rendle, S. 2010. Factorization Machines. In *2010 IEEE International Conference on Data Mining*, 995–1000.

Stone, M. 1948. The generalized weierstrass approximation theorem. *Mathematics Magazine*, 21(5): 237–254.

Udrescu, S.-M.; and Tegmark, M. 2020. Ai feynman: A physics-inspired method for symbolic regression. *Science Advances*, 6(16).

Weierstrass, K. 1885. On the analytical representability of so-called arbitrary functions of a real variable. *Meeting reports of the Royal Prussian Academy of Sciences in Berlin*, 2: 633–639.

Wilson, A.; and Adams, R. 2013. Gaussian Process Kernels for Pattern Discovery and Extrapolation. In Dasgupta, S.; and McAllester, D., eds., *Proceedings of the 30th International Conference on Machine Learning*, volume 28-3 of *Proceedings of Machine Learning Research*, 1067–1075. Atlanta, Georgia, USA: PMLR.

Wu, Y.; Zhu, Z.; Liu, F.; Chrysos, G. G.; and Cevher, V. 2022. Extrapolation and Spectral Bias of Neural Nets with Hadamard Product: a Polynomial Net Study. *36th Conference on Neural Information Processing Systems*.

Xioufis, E. S.; Groves, W.; Tsoumakas, G.; and Vlahavas, I. P. 2012. Multi-Label Classification Methods for Multi-Target Regression. https://arxiv.org/abs/1211.6581v1. Accessed: 2023-12-31.

Yang, Y.; Hou, M.; and Luo, J. 2018. A novel improved extreme learning machine algorithm in solving ordinary differential equations by Legendre neural network methods. *Advances in Difference Equations*, 4(1): 469.

Zhang, H.; Nettleton, D.; and Zhu, Z. 2019. Regression-Enhanced Random Forests. https://arxiv.org/abs/1904.10416. Accessed: 2023-12-31.

Zjavka, L. 2011. Differential polynomial neural network. *Journal of Artificial Intelligence*, 4 (1): 89–99.