Big Data and Information Analytics

http://www.aimspress.com/journal/BDIA

BDIA, 3(2): 41–53.
DOI:10.3934/bdia.2018006
Received: 09 November 2018
Accepted: 23 November 2018
Published: 06 December 2018

Research article

Comparing theory based and higher-order reduced models for fusion simulation data

David E. Bernholdt[1], Mark R. Cianciosa[2], David L. Green[2], Kody J.H. Law[3,*], Alexander Litvinenko[4] and Jin M. Park[5]

[1] Computer Science and Mathematics Division, Oak Ridge National Laboratory, Oak Ridge, TN, USA

[2] Fusion Division, Oak Ridge National Laboratory, Oak Ridge, TN, USA

[3] School of Mathematics, University of Manchester, UK

[4] Extreme Computing Research Center, King Abdullah University of Science and Technology, Thuwal, Saudi Arabia

[5] General Atomics, San Diego, CA, USA

* **Correspondence:** Email: kody.law@manchester.ac.uk; Tel: +44(0)1613055901.

Abstract: We consider using regression to fit a theory-based log-linear ansatz, as well as higher order approximations, for the thermal energy confinement of a Tokamak as a function of device features. We use general linear models based on total order polynomials, as well as deep neural networks. The results indicate that the theory-based model fits the data almost as well as the more sophisticated machines, within the support of the data set. The conclusion we arrive at is that only negligible improvements can be made to the theoretical model, for input data of this type.

Keywords: general linear models; polynomial models; deep neural networks; plasma fusion; Tokamak

1. Setup

Assume we have a set of labelled data points, or training data pairs $\mathcal{D} = \{x^{(i)}, y^{(i)}\}_{i=1}^N$, where $x^{(i)} \in \mathbb{R}^K$ and $y^{(i)} \in \mathbb{R}_+$.

We postulate amongst a family of ansatz $f(\cdot;\theta)$, parameterized by $\theta \in \mathbb{R}^P$, that if we can find a θ^* such that for all $i = 1, \ldots, N$,

$$y^{(i)} \approx f(x^{(i)}; \theta^*),$$

then for all x' within the convex hull of the set $\{x^{(i)}\}_{i=1}^N$, and ideally in a high-probability region over training data distribution, we will have

$$y' \approx f(x'; \theta^*)\,,$$

where y' is the true output corresponding to input x'. In particular, the hope is that if we derive f based on physical theory, then in fact it will hold that even for x' different from the set $\{x^{(i)}\}_{i=1}^N$.

Here we consider parametric approximations, given by generalized linear models (GLM), in particular polynomial models, as well as nonlinear models (NM) given by neural networks (NN).

We will consider the problem of finding the θ^* which minimizes data misfit

$$\Phi(\theta) = \sum_{i=1}^{N} \ell(y^{(i)}, f(x^{(i)}; \theta))\,,$$

for some distance function ℓ. Here we use the standard choice $\ell(\cdot) := |\cdot|^2 := \|\cdot\|_2^2$. It makes sense to interpret this as the log-likelihood of the data given the parameter, as a function of the parameter.

2. Physical setup and theory-based model

In recent work [9], a theory-based scaling of thermal energy confinement time has been derived based on a comprehensive turbulent transport model trapped gyrokinetic Landau fluid (TGLF) [13] in core coupled to the edge pedestal (EPED) [12] model, especially in burning plasma conditions with dominant fusion alpha particle heating for future reactor design. The simulation dataset consists of a massive number of predictive Integrated Plasma Simulator (IPS) FASTRAN [8] simulations, self-consistent with core transport, edge pedestal, fusion alpha particle heating, and MHD equilibrium, built upon a modern integrated modeling framework, IPS.

The $K = 9$ input features are $x_1 = R$ = major radius, $x_2 = a$ = minor radius, $x_3 = \kappa = V/2\pi^2a^2R$, where V = plasma volume, $x_4 = \delta$ = triangularity, $x_5 = I_p$ = plasma current, $x_6 = b_0$ = toroidal magnetic field, $x_7 = \bar{n}_e$ = line average density, $x_8 = Z_{\rm eff}$ =, $x_9 = P_{\rm loss}$ = loss power. The output $y = \tau$ is the thermal energy confinement time.

For input features $x^{(i)} \in \mathbb{R}^K$, $i = 1, \ldots, N$, the theoretical relationship with the output thermal energy confinement time $y^{(i)} \in \mathbb{R}_+$, is given by the following ansatz

$$f_{\rm th}(x; \theta) = \theta_0 \prod_{k=1}^{K} x_k^{\theta_k} \tag{2.1}$$

or on a logarithmic scale (redefining $\log \theta_0 \to \theta_0$)

$$\log f_{\text{th}}(x;\theta) = \theta_0 + \sum_{k=1}^{K} \theta_k \log(x_k) \tag{2.2}$$

3. Parametric regression

Parametric models afford the luxury of discarding the data set after training, and reduce subsequent evaluations of the model to a cost relating to the number of parameters only [1, 7, 10]. However, these models are much more rigid than the non-parametric ones, by virtue of restricting to a particular parametric family.

3.1. General linear models

The simplest type of parametrization is in terms of coefficients of expansion in some basis $\{\psi_i\}_{i=0}^{P-1}$. Here we choose appropriately ordered monomials with total degree $\leq p$, then the number of parameters is $P = (K+p)!/K!p!$. The polynomials $\{\psi_i\}$ are products of monomials in the individual variables, such as $\psi_i(x) = \prod_{k=1}^{K} \psi^{\alpha_k(i)}(x_k)$, where $\psi_0(x) = 1$, each index $i \in \mathbb{Z}_+$ is associated to a unique multi-index $\alpha(i) = (\alpha_1(i), \ldots, \alpha_K(i)) \in \mathbb{Z}_+^K$, and the single variable monomials are defined as $\psi^{\alpha_k(i)}(z) = z^{\alpha_k(i)}$ for $z \in \mathbb{R}$. In particular, for $i = 1, \ldots, K$, $\alpha(i)_i = i$ and $\alpha(i)_j = 0$ for $j \neq i$, i.e. $\psi_i(x) = x_i$. The polynomials are constructed with sglib (https://github.com/ezander/sglib) and the computations are done with Matlab.

Assume that $f(\cdot;\theta) = \sum_{i=0}^{P-1} \theta_i \psi_i(\cdot)$. Under the assumption that the data is corrupted by additive Gaussian white noise, the negative log-likelihood (i.e. data misfit) takes the form

$$\Phi(\theta) = \sum_{i=1}^{N} |f(x^{(i)};\theta) - y^{(i)}|^2 \tag{3.1}$$

Let $X = [x^{(1)}, \ldots, x^{(N)}]^T$ and $Y = [y^{(1)}, \ldots, y^{(N)}]^T$, and let $\Psi(X)_{ij} = \psi_j(x^{(i)})$. Then

$$\Phi(\theta) = |\Psi(X)\theta - Y|^2$$

and the maximum likelihood solution $\theta^* \in \mathbb{R}^P$ is given by

$$\theta^* = \left(\Psi(X)^T \Psi(X)\right)^{-1} \Psi(X)^T Y \,.$$

Predictions of outputs Y' for some X' are later given by

$$Y' = \Psi(X')\theta^* \,.$$

We will also consider log-polynomial regression, where the monomials are composed with log, i.e. the basis functions become $\psi_i \circ \log$, where log acts component-wise. We then define $\log f(\cdot;\theta) = \sum_{i=0}^{P-1} \theta_i \psi_i(\log(\cdot))$ and the misfit is

$$\Phi(\theta) = \sum_{i=1}^{N} |\log f(x^{(i)};\theta) - \log y^{(i)}|^2 \,. \tag{3.2}$$

This is the negative log-likelihood under a multiplicative lognormal noise assumption, as opposed to the additive Gaussian noise assumption of (3.1). To understand the motivation for this, observe that (2.2) is a log-linear model corresponding to the logarithm of (2.1). Therefore, this setup includes the theoretical model prediction, when $p = 1$, and provides a way to systematically improve upon the theoretical model, by increasing p.

As a final note, as p increases, there is a chance of including more irrelevant parameters, in addition to ones which may be relevant. Therefore, we must regularize somehow. For the GLM, it is natural to adopt a standard L^2 Tikhonov regularization term $\lambda|\theta|^2$, resulting in the following modified objective function, for $\lambda > 0$,

$$\bar{\Phi}(\theta) = \Phi(\theta) + \lambda|\theta|^2 \tag{3.3}$$

which again can be exactly solved

$$\theta^* = \left(\Psi(X)^T\Psi(X) + \lambda I\right)^{-1} \Psi(X)^T Y$$

where I is the $P \times P$ identity matrix. One can readily observe that this provides an upper bound to the pseudo inverse operator mapping the observation set to the optimal parameter (in other words $\Psi(X)^T\Psi(X) + \lambda I$ has a spectrum lower bounded by λ).

3.2. Nonlinear models

In some very particular cases, it may make sense to invoke nonlinear models, at the significant additional expense of solving a nonlinear least squares problem (in the easiest instance) as opposed to the linear least squares solution presented above. The most notable example is deep neural networks. It has been shown that these models perform very well for tasks such as handwritten digit classification, voice and object recognition [3], and there has recently even been mathematical justification that substantiates this [5]. However, [6] show that one can sometimes find very small perturbations on input values which lead to misclassification. The methods can also be used for regression problems, and will be considered below.

3.2.1. Deep neural networks

Let $\theta_i \in \mathbb{R}^{P_i}$ ($P = \sum_{i=1}^{L} P_i$, where L is the number of layers) be parameters and g_i be some possibly nonlinear function, corresponding to layer i, and let

$$f(\cdot;\theta) = g_L(\cdots g_2(g_1(\cdot;\theta_1);\theta_2);\cdots;\theta_L).$$

As an example, $\theta_i = (A_i, b_i)$ is typically split into parameters $A_i \in \mathbb{R}^{n_i \times n_{i-1}}$, $b_i \in \mathbb{R}^{n_i}$, defining an affine transformation $A_i x + b_i$, and $g_i(x;\theta_i) = \sigma_i(A_i x + b_i)$, where σ_i is a simple nonlinear "activation function" which is applied element-wise. In this case n_i are the number of auxiliary variables, or "neurons", on level i, $n_0 = K$ is the input dimension, $n_L = 1$ is the output dimension, and there are L levels. For regression one typically takes σ_L =Id the identity map, as will be done here.

For the neural networks used here, $\sigma_i = \sigma : \mathbb{R} \to \mathbb{R}_+$, for $i < L$, is the the rectified linear unit (ReLU) defined by

$$\sigma(z_i) = \max\{0, z_i\} \tag{3.4}$$

This activation function contains no trainable parameters, and we iterate that it is applied element-wise to yield a function $\mathbb{R}^{n_i} \to \mathbb{R}^{n_i}_+$.

The misfit again takes the form

$$\Phi(\theta) = \sum_{i=1}^{N} |f(x^{(i)};\theta) - y^{(i)}|^2 \tag{3.5}$$

except now we have a nonlinear and typically non-convex optimization problem.

There are many methods that can be employed to try to find the minimum of this objective function, and they will generally be successful in finding some minimum (though perhaps not global). Due to the mathematically simple functions that make up the neural network, derivatives with respect to the unknown parameters can be determined analytically. A popular method to solve non-convex optimization problems is stochastic gradient descent, or Robbins-Monro stochastic approximation algorithm [4, 11], which is defined as follows. Let $\widehat{\nabla\Phi}$ be an unbiased estimate of $\nabla\Phi$, i.e. a random variable such that $\mathbb{E}\widehat{\nabla\Phi}\,(\theta) = \nabla\Phi\,(\theta)$. Then let

$$\theta_{i+1} = \theta_i - \gamma_i \widehat{\nabla\Phi}\,(\theta_i) \tag{3.6}$$

If $\{\gamma_i\}$ are chosen such that $\sum_i \gamma_i = \infty$ and $\sum_i \gamma_i^2 < \infty$, then the algorithm is guaranteed to converge to a minimizer [4, 11], which is global if the function is convex [2]. Noting that we can compute

$$\nabla\Phi(\theta) = \sum_{i=1}^{N} \nabla_\theta \ell(f(x^{(i)};\theta), y^{(i)})$$

easily in this case, one could use for example $\widehat{\nabla\Phi}(\theta) = \nabla\Phi(\theta) + \xi$, where $\xi \in \mathbb{R}^P$ is some random variable, for example a Gaussian, with $\mathbb{E}\xi = 0$. Something more clever and natural can be done in the particular case of (3.5). Recall that N can be very large. Observe that (3.5) is, up to a multiple of N, an equally weighted average of the summands $\ell(f(x^{(i)};\theta), y^{(i)})$. Let $\{n_i\}_{i=1}^{N_{\text{sub}}}$ be a random (or deterministic) subset of $N_{\text{sub}} < N$ indices, where N_{sub} could be 1. Then the following provides an unbiased estimator which is also much cheaper than the original gradient (since it uses a subset of the data)

$$\widehat{\nabla\Phi}(\theta) := \frac{N}{N_{\text{sub}}} \sum_{i=1}^{N_{\text{sub}}} \nabla_\theta \ell(f(x^{(n_i)};\theta), y^{(n_i)})\,.$$

Naturally the variance of this estimator will affect the convergence time of the algorithm and we refer the interested reader to the monograph [4] for discussion of the finer points of this family of algorithms. Neural Network models were implemented and trained using the Mathematica Neural Network functions built on top of the MXNet framework.

Increasing the number of free parameters, by increasing the depth or width of the network, allows more flexibility and can provide a better approximation to complex input output relationships.

However, as with GLM, increasing the number of degrees of freedom in the optimization problem also increases the susceptibility of the algorithm to instability from fitting noise in the data too accurately. A Tikhonov regularization could be employed again, but here we appeal to another strategy which is applicable only to DNN. In particular, the training data is randomly divided into N training samples and $N_{\rm val}$ validation samples. The algorithm 3.6 is run with the training data, however the loss of the validation data is computed along the way

$$L(\theta) = \sum_{i=N+1}^{N+N_{\rm val}} |f(x^{(i)};\theta) - y^{(i)}|^2 ,$$

and the network with the smallest loss on the validation data is retained at the end. This serves as a regularization, i.e. avoids fitting noise, since the training and validation sets are corrupted by different realizations of noise.

4. Numerical results

We consider a data set of 2822 data points with $K = 9$ input features and a single output. A subset of $N = 2100$ data points will be used for training, with the rest of the $N_{\rm out}$ data points aside for testing. General linear and log linear models with order up to 6 are considered, as well as deep neural networks.

4.1. GLM

An empirically chosen regularization of $\lambda = 1$ for $p = 2$, $\lambda = 50$ for $p = 3$, and $\lambda = 100$ for $p > 3$ is used in (3.3). Figure 1 presents the results for the theoretical log linear model, as well as the best fit based on a expansion in log polynomials of total order 6, for the out-of-sample set of testing data. It is clear already that the enriched model set brings negligible improvement in the reconstruction fidelity. The left subpanels of Figure 4 show some slices over the various covariates, where the other covariates are fixed at a value from the center of the data set. This gives some idea of how the reconstruction behaves, although one should bear in mind that the data set forms a manifold and not a hypercube, in $\mathbb{R}^9$, and so these slices are bound to include unobserved subsets of the parameter space. Furthermore, there may be sparse coverage even where there is data. This can be observed also in Figure 4 which illustrates the single component marginal histograms over the data.

Figure 2 show the relative L^2 fidelity and the coefficient of determination R^2, defined below, for log polynomial and polynomial models of increasing order.

The relative error $L^2_{\rm rel}(f)$ of the surrogate f is given by

$$L^2_{\rm rel}(f) := \sqrt{\frac{\sum_{i=N+1}^{N+N_{\rm out}} |f(x^{(i)};\theta^*) - y^{(i)}|^2}{\sum_{i=N+1}^{N+N_{\rm out}} |y^{(i)}|^2}} .$$

The coefficient of determination $\mathrm{R}^2(f)$ is given by

$$\mathrm{R}^2(f) := 1 - \frac{\sum_{i=N+1}^{N+N_{\rm out}} |f(x^{(i)};\theta^*) - y^{(i)}|^2}{\sum_{i=N+1}^{N+N_{\rm out}} |y^{(i)} - \overline{y}|^2} ,$$

where

$$y = \frac{1}{N_{\text{out}}} \sum_{i=N+1}^{N+N_{\text{out}}} y^{(i)}.$$

Both models begin to saturate at the same level of fidelity, indicating that this is a fundamental limitation of the information in the data. Furthermore, one observes from Figure 2 that the saturation level of the fidelity occurs with only a few percentage points of improvement over the theoretically derived log-linear model.

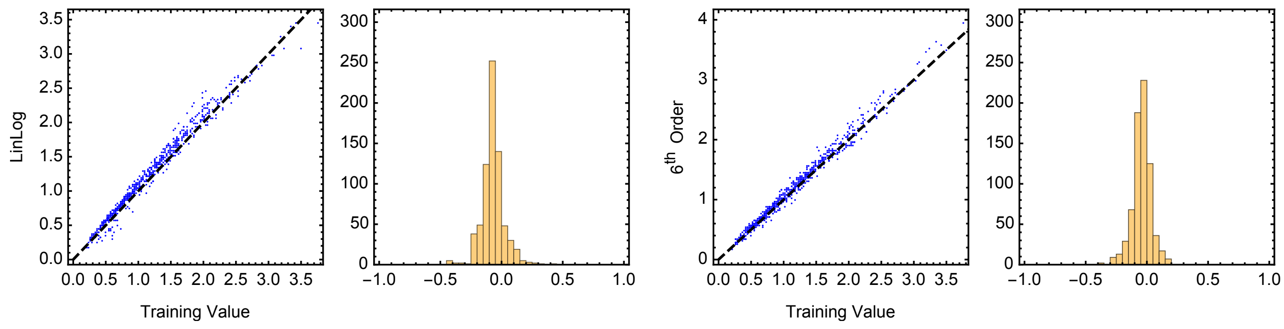

Figure 1. Comparison of true values with GLM prediction, for log linear (left), and log sixth order (right). The right subplots show the error distribution histogram (difference between values of the truth and the prediction). Out-of-sample, test data.

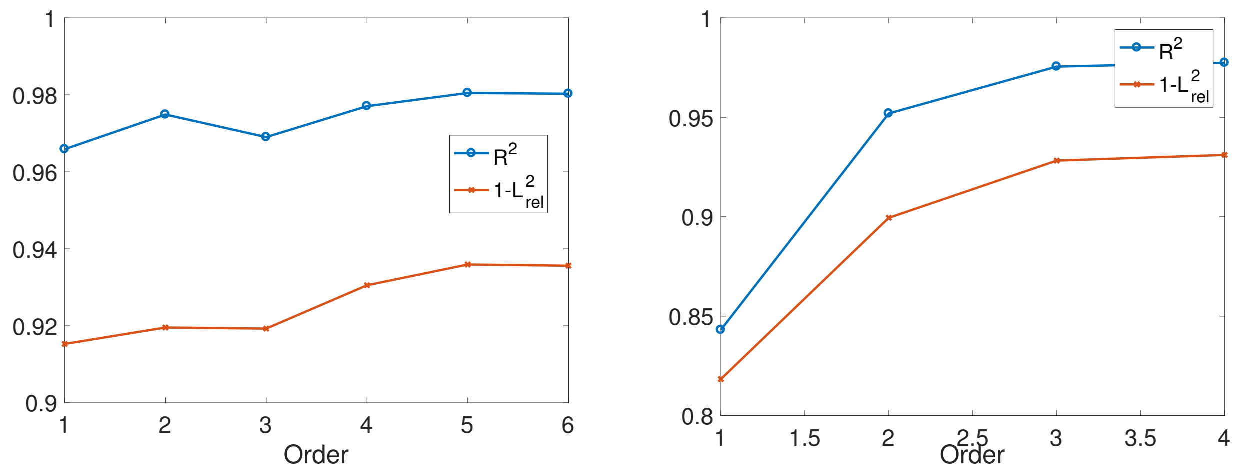

Figure 2. Relative L^2 error and coefficient of determination R^2 for out-of-sample test data as a function of model complexity, for log polynomial models (left) and direct polynomial models (right).

4.2. Neural networks

Table 1. Description of the 3 layer DNN.

Layer Type	Inputs	Width	Weights	Biases	Total Parameters
Linear	9	20	180	20	200
ReLU	20	20	0	0	0
Linear	20	20	400	20	420
ReLU	20	20	0	0	0
Linear	20	1	20	1	21
				Total	641

The neural network models were setup and trained using the built in neural network functions of Mathematica, which is built around the MXNet machine learning framework. The $N_{\text{val}} = N_{\text{out}}$ samples are used in this case for validation, as described in the end of subsection 3.2.1.

By connecting various combinations of layers, the neural network model gains the flexibility to model the behavior of the training data. As the size of the network increases so do the number of unknown parameters in the model. If there is too much flexibility then the network model begins to conform to the noise in the data. In addition, due to the nature of gradient decent methods, the minima reached is a local minima dependent on the starting position. As the number of unknown parameters increases, in particular if it exceeds the amount of training data, one expects the inversion to become unstable, with an increased risk of converging to an undesirable local minimum.

To reduce the degeneracy of the objective function, the total number tunable weights and biases in the network will be kept below N, although this can be relaxed if a Tikhonov or other type of regularization is employed. Table 1, shows the make up of the neural network model used here. Initial weights and biases and chosen at random. The networks will be trained multiple times with random initial parameters to explore various local modes. Three examples are shown in Figures 3, 4. Figure 3 shows four pairs of images, with the left being the output in comparison to the true value, and the right being a histogram of the difference between the 2. The left two pairs and the top right are all corresponding to different instances of the same network optimization, but with different random selections of training data and different initial guesses. The bottom right is the best fit log linear model 2.2.

The L^2_{rel} and R^2 fidelity measures are shown in Table 2. What one observes here is that one finds almost no improvement at all over the theoretical log linear model with this $P = 641$ parameter neural network, while in fact we do manage to improve the accuracy by a few percent with higher order GLM using $P = 5005$ parameters (see Figure 2). Figure 4 shows 9 pairs of panels, one for each parameter. This indicates that the models perform better where the data is more dense. Also, the higher order polynomial models can behave wildly outside the support of the data. See for example the left bottom 3 panels.

Finally, Figure 5 shows the pairwise marginal histograms, which gives a slightly better indication of the shape of the data distribution.

Table 2. Total loss of all available data.

Network	Validation loss	$1 - L_{\text{rel}}^2$	R^2
3 Layer 1	0.0148	0.915	0.963
3 Layer 2	0.0146	0.917	0.964
3 Layer 3	0.0140	0.918	0.965
Linlog	0.0161	0.914	0.960

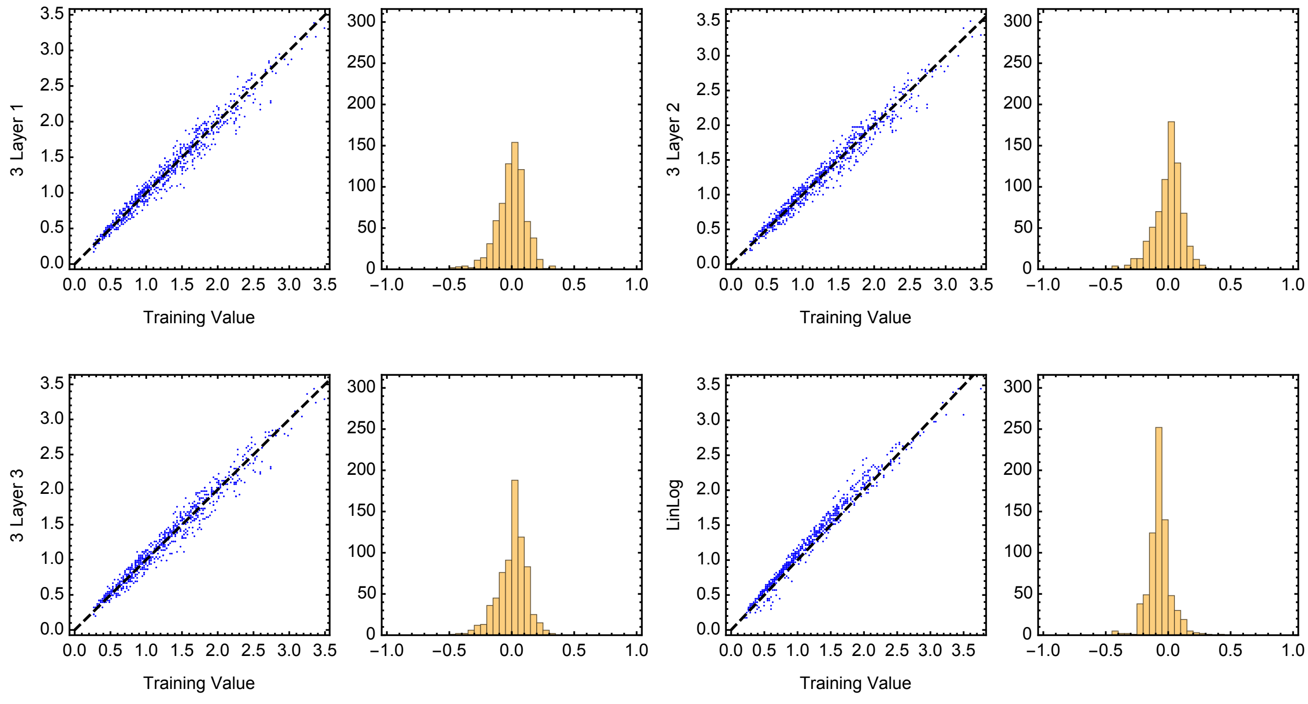

Figure 3. Histograms of the 3 DNN instances and the loglinear model. The right subplots show the error distribution histogram (difference between values of the truth and the prediction).

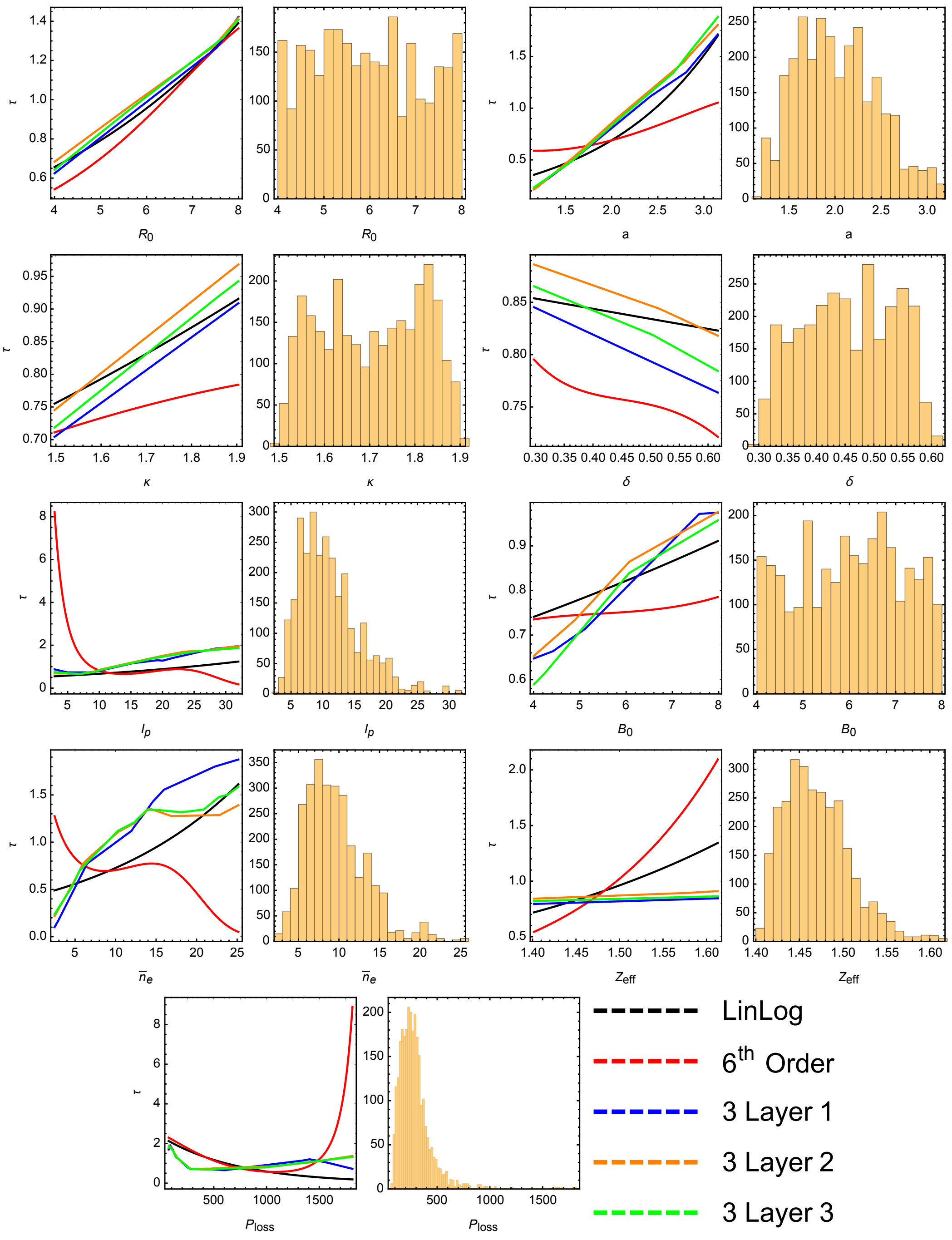

Figure 4. Scaling and input data distribution log linear, sixth order, and NN models. The left subplots show a slice through the parameter space of the various models, giving an indication of their behaviour relative to one another. The right subplots show the marginal histograms for the given parameter. One can observe a large deviation between the curves where the data is quite sparse, particularly for the higher order polynomial models.

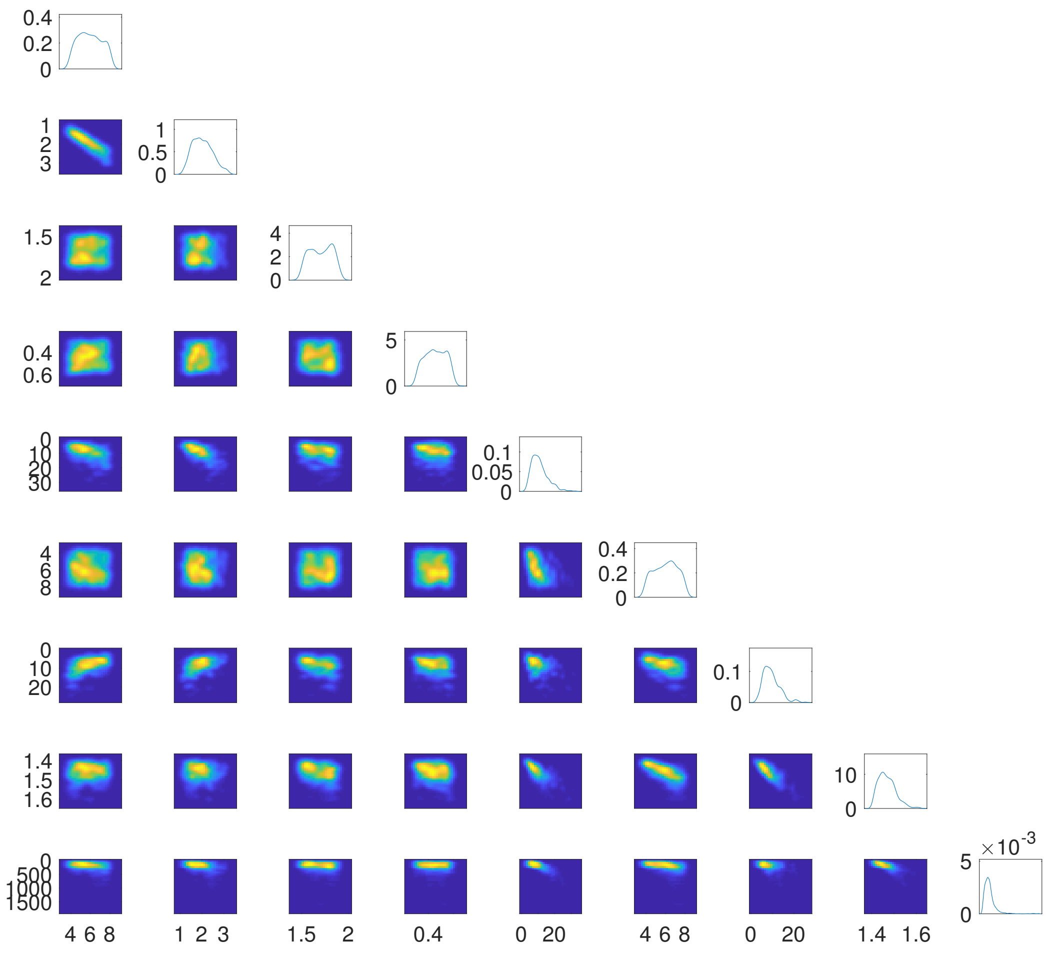

Figure 5. Pairwise marginal histograms of the input data X. This gives some idea of the pairwise correlations between the different features over the input data.

5. Summary

In this work we considered fitting a theory-based log-linear ansatz, as well as higher order approximations for the thermal energy confinement of a Tokamak. It is a supervised machine learning regression problem to identify the map $\mathbb{R}^K \to \mathbb{R}_+$ with $K = 9$ inputs. We parametrized the map using general linear models based on total order polynomials up to degree 6 ($P = 5005$ parameters), as well as deep neural networks with up to $P = 641$ parameters. The results indicate that the theory-based model fits the data almost as well as the more sophisticated machines, within the support of the data set. The conclusion we arrive at is that only negligible improvements can be made to the theoretical model, for input data of this type. Further work includes (i) Bayesian formulation of the inversion for uncertainty quantification (UQ), or perhaps derivative-based UQ (sensitivity), (ii) inclusion of further hyper-parameters in the GLM for improved regularization within an empirical Bayesian framework, (iii) using other types of regularization for the deep neural network, in order to allow for more parameters, and (iv) exploring alternative machine learning methods, such as Gaussian processes, which naturally include UQ.

Acknowledgments

This work has been supported by the U.S. Department of Energy, Office of Science, Office of Fusion Energy Sciences, using the DIII-D National Fusion Facility, a DOE Office of Science user facility under awards, DE-FG02-04ER54761 and DE-FC02-04ER54698.

Conflict of interest

All authors declare no additional conflicts of interest in this paper.

References

1. Christopher MB (2006) Pattern recognition and machine learning. *J Electron Imaging* 16: 140–155.
2. Boyd S, Vandenberghe V (2004) Convex optimization, Cambridge university press.
3. Goodfellow I, Bengio Y, Courville A (2016) Deep Learning, MIT Press.
4. Kushner H, Yin GG (2003) Stochastic approximation and recursive algorithms and applications, Springer Science & Business Media, 35.
5. Mallat S (2016) Understanding deep convolutional networks. *Phil Trans R Soc A* 374: 20150203.
6. Moosavi-Dezfooli SM, Fawzi A, Frossard P (2016) Deepfool: A simple and accurate method to fool deep neural networks, In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2574–2582.
7. Murphy KP (2014) Machine learning: A probabilistic perspective. *CHANCE* 27: 62–63.
8. Park JM, Ferron JR , Holcomb CT, et al. (2018) Integrated modeling of high βn steady state scenario on diii-d. *Phys Plasmas* 25: 012506.
9. Park JM, Staebler G, Snyder PB, et al. (2018) Theory-based scaling of energy confinement time for future reactor design. Available from: http://ocs.ciemat.es/EPS2018ABS/pdf/P5.1096.pdf.
10. Rasmussen CE (2003) Gaussian processes in machine learning, MIT Press.
11. Robbins H, Monro S (1951) A stochastic approximation method. *Ann Math Stat*: 400–407.
12. Snyder PB, Burrell KH, Wilson HR, et al. (2007) Stability and dynamics of the edge pedestal in the low collisionality regime: Physics mechanisms for steady-state elm-free operation. *Nucl Fusion* 47: 961.
13. Staebler GM, Kinsey JE, Waltz RE (2007) A theory-based transport model with comprehensive physics. *Phys Plasmas* 14: 055909.

AIMS Press