entropy

MDPI

Article

Predicting the Critical Number of Layers for Hierarchical Support Vector Regression

Ryan Mohr [1,†], Maria Fonoberova [1,†], Zlatko Drmač [2,*,†], Iva Manojlović [1,†] and Igor Mezić [1,3,†]

[1] AIMdyn Inc., Santa Barbara, CA 93106, USA; mohrr@aimdyn.com (R.M.); mfonoberova@aimdyn.com (M.F.); imanojlovic@aimdyn.com (I.M.); mezici@aimdyn.com (I.M.)
[2] Faculty of Science, University of Zagreb, 10000 Zagreb, Croatia
[3] Department of Mechanical Engineering, University of California, Santa Barbara, CA 93106, USA
* Correspondence: drmac@math.hr
† These authors contributed equally to this work.

Abstract: Hierarchical support vector regression (HSVR) models a function from data as a linear combination of SVR models at a range of scales, starting at a coarse scale and moving to finer scales as the hierarchy continues. In the original formulation of HSVR, there were no rules for choosing the depth of the model. In this paper, we observe in a number of models a phase transition in the training error—the error remains relatively constant as layers are added, until a critical scale is passed, at which point the training error drops close to zero and remains nearly constant for added layers. We introduce a method to predict this critical scale a priori with the prediction based on the support of either a Fourier transform of the data or the Dynamic Mode Decomposition (DMD) spectrum. This allows us to determine the required number of layers prior to training any models.

Keywords: support vector regression; fourier transform; dynamic mode decomposition; koopman operator

Citation: Mohr, R.; Fonoberova, M.; Drmač, Z.; Manojlović, I.; Mezić, I. Predicting the Critical Number of Layers for Hierarchical Support Vector Regression. *Entropy* **2021**, *23*, 37. https://doi.org/10.3390/e23010037

Received: 1 December 2020
Accepted: 22 December 2020
Published: 29 December 2020

1. Introduction

Many of the machine learning algorithms require the correct choice of hyperparameters to give the best description of the given data. One of the most popular methods for choosing hyperparameters is doing grid-search. In grid search, the model is trained for some points in hyperparameter space which are called a grid and then the model with the lowest error on the validation set is chosen. Other methods use alternative optimization algorithms, such as genetic algorithms. All these methods include training model and calculating the error multiple times, which can be very expensive if the hyperparameter space is large.

The purpose of this paper is to develop a method for determining hyperparameters for Support Vector Regression (SVR) with Gaussian kernels from time-series data only. Unlike other approaches for choosing hyperparameters, our method identifies a set of hyperparameters without needing to train models and perform a grid search (or executing some other hyperparameter optimization algorithm), thereby bypassing a potentially costly step. The proposed method identifies the inherent scale and complexity of the data and adapts the SVR accordingly. In particular, we give a method for determining the scale of Gaussian kernel by connecting it to the most important frequencies of the signal, as determined by either a Fourier transform or Dynamic Mode Decomposition. Thus we leverage classical and generalized harmonic analysis to inform the choice of hyperparameters in modern machine learning algorithms. A pertinent question is why not just use FFT? The answer is that HSVR are better suited to model strongly locally varying data whereas, for FFT to be efficient, the data need to possess some symmetry such as space or time translation.

Previous Work

There are many different approaches in tuning SVR hyperparameters for reducing the generalization error. Some popular methods of estimating generalization error are Leave-one-out (LOO) score and k cross-validation score. They are easy to implement, but for calculating those measures more models have to be trained for each combination of hyperparameters that need to be tested. This can be prohibitively expensive, so other error estimates, which are easier to calculate, were developed. These methods include the Xi-Alpha bound [1], the generalized approximate cross-validation [2], the approximate span bound [3], the VC bound [3], the radius-margin bound [3] or the quality functional of the kernel [4].

For choosing hyperparameters, the simplest method is to perform grid search over the space of hyperparameters and then choose the ones with the lowest error estimation. However, grid search suffers from the curse of dimensionality, scaling exponentially with the dimension of the hyperparameter configuration space. Other work in hyperparameter optimization (HPO) seek to mitigate this problem. Random search samples the configuration space and can serve as a good baseline [5,6]. Bayesian optimization techniques to find optimal hyperparameters, often using Gaussian processes as surrogate functions, offers a more more computational efficient algorithm than grid search or random search requiring fewer attempts to find the optimal parameters [6]. However, Bayesian optimization in this form requires more computational resources [6].

There are also gradient-based approaches [7–9]. There are also several derivative-free optimization methods. For example, in [10], a pattern search methodology for hyperparameters, the parameter optimization method based on simulated annealing [11], Bayesian method based on MCMC for estimating kernel parameters [12], and also methods based on Genetic algorithms [13–15].

What all of these methods have in common is that they require training a model and evaluating the error for each parameter set that needs to be tried. This can be quite expensive, both in time and computational resources, if there are a lot of data or we want to train multiple models at the same time. They are iterative, which means that we usually do not know how many models will be trained before getting an estimation of the best hyperparameters. There are methods, such as early cutoff, that try to reduce these burdens, but fundamentally a model needs to be trained for each set of candidate hyperparameters.

In contrast, our approach gives a set of hypeparameters without ever computing a single model. Specifically, in this paper, we are interested in modeling multiscale signals with a linear combination of SVRs. Two fundamental questions are (1) how many SVR models are going to be used? and (2) what should the scale be for each SVR model. Clearly the dimension of the configuration space for the scale parameters is conditional on the number of layers. Our methods are built off fast spectral methods like fast Fourier transform and Dynamic Mode Decomposition and return both the number of layers (number of SVR models) and the scales for each SVR without ever having to train a model. Bypassing the expensive step of computing a model for each candidate set of hyperparameters can lead to dramatic savings in computational time and resources.

2. Methods

2.1. Support Vector Regression

Support Vector Machines (SVM) have been introduced in [16] as a method for classification. The goal was to set hyperplane between classes, where hyperplane is defined by linear combination of a subset of a training set, called Support Vectors. The problem is formulated as a quadratic optimization problem that is convex so it has a unique solution. The problem with SVM is that it can only find a linear boundary between classes which is often not possible. The trick is to map the training data to a higher dimensional space and then use kernel functions to represent the inner product of two data vectors projected onto this space. Then only inner products are needed for finding the best parameters. The ad-

vantage of this approach is that we can implicitly map data onto infinite dimensional space. One of the most used kernels is Gaussian function defined by

$$k(x, x') = \exp\left(-\frac{\|x - x'\|_2^2}{\sigma^2}\right) = \exp(-\gamma\|x - x'\|_2^2), \tag{1}$$

where x and x' are n-dimensional feature vectors, σ^2 is the variance of the Gaussian function, and $\gamma = 1/\sigma^2$ is the scale parameter that is usually specified as an input parameter to SVR toolboxes.

The SVM approach has been extended in [17] to regression problems and it is called Support Vector Regression (SVR). Let $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ be the training set, where x_i is vector in input space $X \subset \mathbf{R}^D$ and $y_i \in \mathbf{R}$ desired output. The aim of SVR is to find a regression function $f : X \to \mathbf{R}$:

$$f(x) = \omega^T \Phi(x) + b, \tag{2}$$

where ω is the weight vector and Φ is a mapping of the data points to a higher-dimensional space and b is threshold constant. ω and b can be found by solving following optimization problem:

$$\min_{\omega,b} \frac{1}{2}\omega^T\omega + C\Sigma_{i=1}^n E_i^+ + C\Sigma_{i=1}^n E_i^-, \tag{3}$$

$$s.t \qquad y_i - \omega^T\Phi(x) - b \leq \epsilon + E_i^+ \tag{4}$$

$$\omega^T\Phi(x) + b - y_i \leq \epsilon + E_i^- \tag{5}$$

$$E_i^+, E_i^- \geq 0, \qquad (i = 1, \ldots, n). \tag{6}$$

The parameter ϵ determines width of tube around the regression curve and points inside it do not contribute to the loss function. Parameter C adjusts the trade off between the regression error and regularization, E^+ and E^- are slack variables for relaxing approximation constraints and measure the distance of each data point from the ϵ tube. In practice, the dual problem is solved, which can be written as:

$$\max_{\alpha^+,\alpha^-} \quad -\frac{1}{2}(\alpha^+ - \alpha^-)K(\alpha^+ - \alpha^-) - \epsilon\sum_{i=1}^n(\alpha_i^+ + \alpha_i^-) + \sum_{i=1}^n y_i(\alpha_i^+ - \alpha_i^-), \tag{7}$$

$$s.t \qquad \sum_{i=1}^n(\alpha_i^+ - \alpha_i^-) = 0 \tag{8}$$

$$\alpha_i^+, \alpha_i^- \in [0, C], \qquad (i = 1, \ldots, n). \tag{9}$$

where $\alpha^+, \alpha^- \in \mathbf{R}^n$ are the dual variables and $K \in \mathbf{R}^{n\times n}$ is the kernel matrix evaluated from a kernel function, $K_{ij} = k(x_i, x_j)$ where $k(x, x')$ is the kernel function. Solving that problem, the regression function becomes:

$$f(x) = \sum_{i=1}^n(\alpha_i^+ - \alpha_i^-)k(x, x_i) + b. \tag{10}$$

Coefficients satisfy following conditions:

$$|\alpha_i^+ - \alpha_i^-| = \begin{cases} 0 & \|y_i - f(x_i)\| < \epsilon \\ \in (0, C) & \|y_i - f(x_i)\| = \epsilon \\ C & \|y_i - f(x_i)\| > \epsilon \end{cases}$$

Data points for which $|\alpha_i^+ - \alpha_i^-|$ is non-zero are called Support Vectors.

2.2. Dynamic Mode Decomposition (Dmd)

Dynamic Mode Decomposition (DMD) was introduced in [18] as a method for extracting dynamic information from flow fields that are either generated by numerical simulation or measured in physical experiment. Rowley et al. connected DMD with Koopman operator theory [19].

Let the data be expressed in a series of snapshots, given by matrix $\mathbf{V_1^N}$:

$$\mathbf{V_0^N} = \{v_0, v_1, \cdots, v_N\}, \tag{11}$$

where $v_i \in \mathbb{R}^m$ stands for the i-th snapshot of the flow field. We assume there exists a linear mapping $\mathbf{A}$ which relates each snapshot v_i to next one v_{i+1},

$$v_{i+1} = \mathbf{A}v_i, \tag{12}$$

and that this mapping is approximately same during each sampling interval, so we approximately have

$$\mathbf{V_0^N} = \{v_0, \mathbf{A}v_0, \cdots, \mathbf{A}^N v_0\}. \tag{13}$$

We assume that characteristics of the system can be described by the spectral information in $\mathbf{A}$. This information is extracted in a data-driven manner using $\mathbf{V_0^N}$. The idea is to use $\mathbf{V_0^N}$ to construct an approximation of $\mathbf{A}$. In [18], this was done as follows. Define $\mathbf{X} = \mathbf{V_0^{N-1}}$ and $\mathbf{Y} = \mathbf{V_1^N}$. Let the singular value decomposition of $\mathbf{X}$ be $\mathbf{X} = \mathbf{U\Sigma V}^*$. Then the representation of a compression of $\mathbf{A}$ is defined as

$$\tilde{\mathbf{A}} = \mathbf{U}^*\mathbf{YV}\Sigma^+ \tag{14}$$

where Σ^+ is the Moore–Penrose pseudo-inverse of Σ. Note that (14) is analytically equivalent to $\mathbf{U}^*\mathbf{AU}$, the compression of $\mathbf{A}$ to the subspace spanned by the columns of $\mathbf{X}$, if $\mathbf{Y} = \mathbf{AX}$. Eigenvectors and eigenvalues of $\mathbf{A}$ are approximations of eigenvectors and eigenvalues of the Koopman operator.

The Koopman operator is an infinite dimensional, linear operator K that acts on all scalar functions g on M as

$$Kg(x) = g(f(x)), \tag{15}$$

where f is a dynamical system such that $x_{k+1} = f(x_k)$. Let λ_j and ϕ_j be eigenvalues and eigenfunctions, i.e.,

$$K\phi_j(x) = \lambda_j\phi_j(x). \tag{16}$$

Let $g(x)$: $M \to \mathcal{R}^p$ be vector of any quantities of interest. If g lies in span of ϕ_j, then it can be written as

$$g(x) = \sum_{j=1}^{\infty} \phi_j(x)v_j. \tag{17}$$

Then we can express $g(x_k)$ as

$$g(x_k) = K^k g(x_0) = K^k \sum_{j=1}^{\infty} \phi_j(x_0)v_j = \sum_{j=1}^{\infty} \lambda_j^k \phi_j(x_0)v_j. \tag{18}$$

The Koopman eigenvalues, λ_j characterize the temporal behaviour of the corresponding Koopman mode v_j, the phase of λ_j determines its frequency, and the magnitude determines the growth rate.

2.3. Hierarchical Support Vector Regression

Classical SVR models which use kernels of a single scale have difficulties approximating multiscale signals. For example, consider the function $f(x) = x + \sin(2\pi x^4)$, for $x \in [0, 2]$, whose graph is given in Figure 1. This function has a continuum of scales. Figure 2 highlights the difficulties that classical single-scale SVR has in modeling such

signals. Using too large a scale σ, the detailed behavior of the data set is not captured [20]; such a model may, however, be useful for a coarse-scale extrapolation outside the training set. Models employing a very small scale σ can capture the training set in detail. However, this makes them very sensitive to noise in the training and severely limits the model's ability to generalize outside the training set [20].

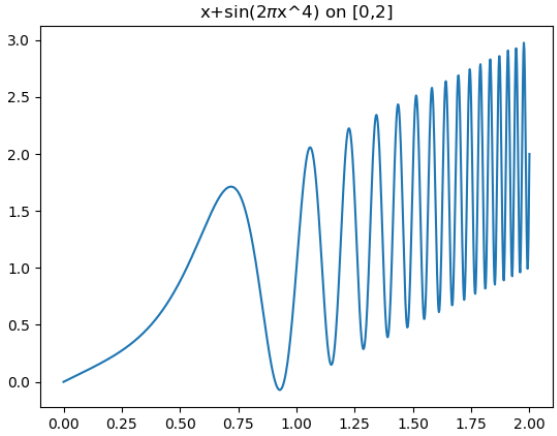
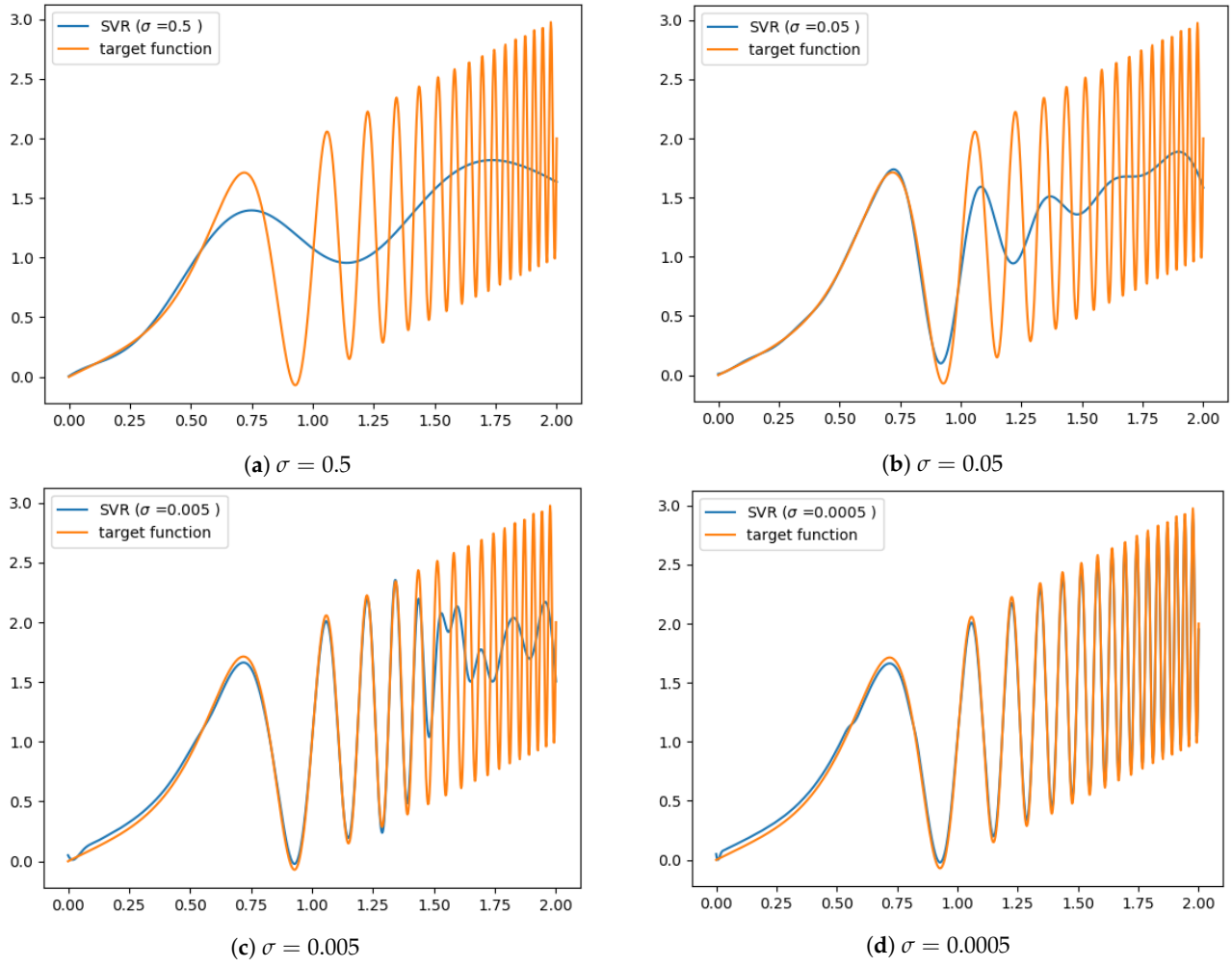


Figure 1. Multiscale example function: $f(x) = x + \sin(2\pi x^4)$.



(**a**) $\sigma = 0.5$ (**b**) $\sigma = 0.05$

(**c**) $\sigma = 0.005$ (**d**) $\sigma = 0.0005$

Figure 2. SVR results with different scale of Gaussian kernel σ = 0.5, 0.05, 0.005, 0.0005. A large kernel provides smooth regression, but cannot reconstruct the details. A small kernel overfits, is unable to generalize, and can be sensitive to noise.

In [20], the authors introduced a multiscale variant of Support Vector Regression which they termed Hierarchical Support Vector Regression (HSVR). The idea behind HSVR is to train multiple SVR models, organized as a hierarchy of layers, each with different scale σ. The HSVR model is then a sum of those individual models, which we write as

$$S(x) = \sum_{\ell=0}^{L} a_\ell(x; \sigma_\ell, \epsilon), \tag{19}$$

where L is number of layers and $a_\ell(x; \sigma_\ell, \epsilon)$ is SVR model on layer ℓ with Gaussian kernel with parameter σ_ℓ. Each SVR layer realizes a reconstruction of the target function at a certain scale. Training the HSVR model precedes from coarser scales to finer scales as follows. Let $\sigma_0 > \sigma_1 > \sigma_L > 0$ be specified. For σ_0, an SVR model $a_0(x; \sigma_0, \epsilon)$ is trained on the signal $f(x)$ (the 0-th residual) and the residual $r_1(x) = f(x) - a_0(x; \sigma_0, \epsilon)$ is computed. We then proceed inductively for $\ell \geq 1$. Given the residual $r_\ell(x)$, train a model $a_\ell(x; \sigma_\ell, \epsilon)$ to approximate it and compute new residual $r_{\ell+1}(x) = r_\ell(x) - a_\ell(x; \sigma_\ell, \epsilon)$. The $(L+1)$-th residual is then

$$r_{L+1}(x) = f(x) - a_0(x; \sigma_0, \epsilon) - \cdots - a_L(x; \sigma_L, \epsilon) = f(x) - S(x). \tag{20}$$

Graphically, the process looks like Figure 3.

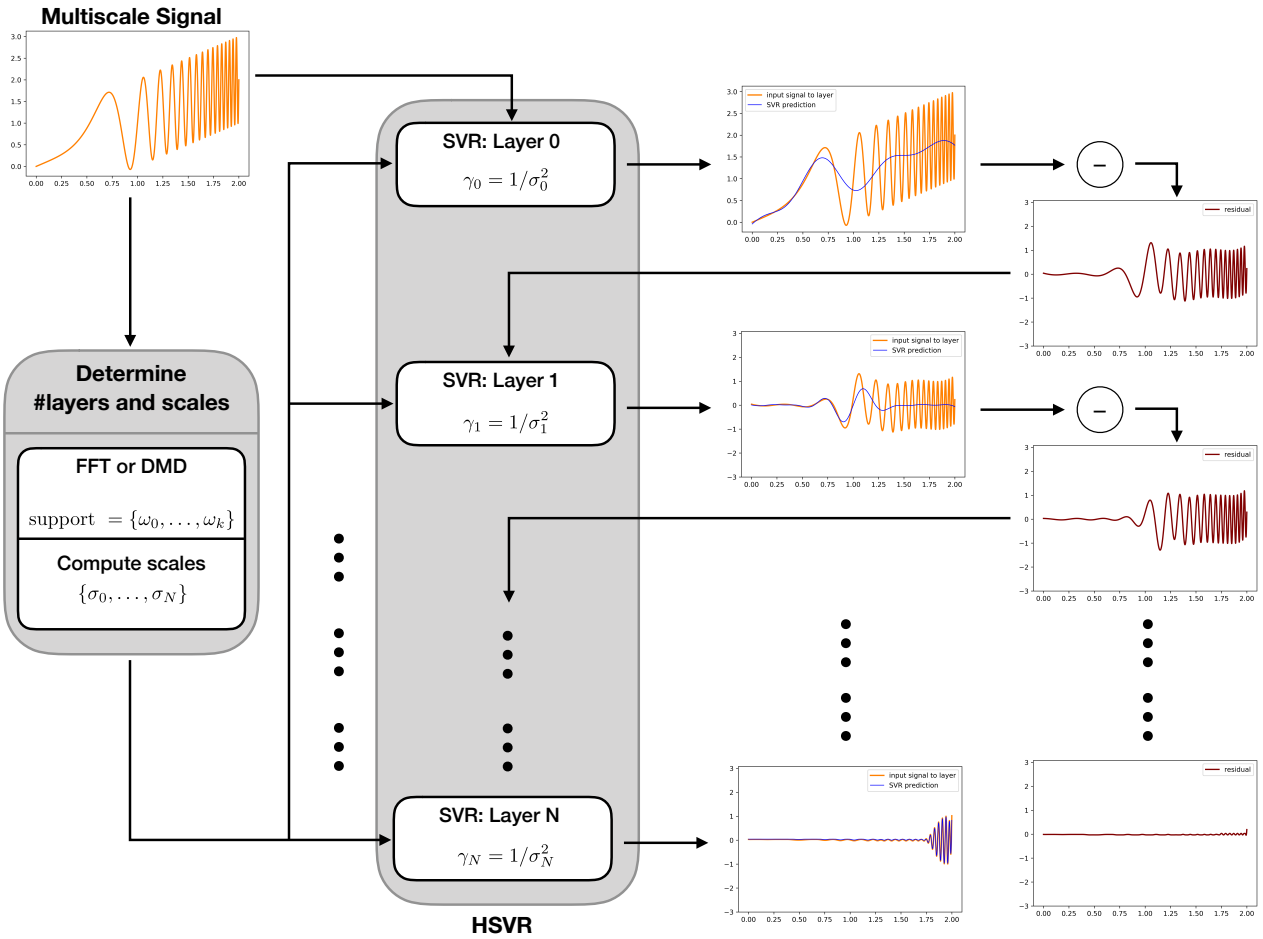


Figure 3. Flowchart of the HSVR modeling process. The input data is first used to compute the scales used for the HSVR model (see Algorithms 1, 2 and 4). At layer 0, an SVR model is trained at the coarsest scale γ_0. The residual is computed by taking the difference between the signal and the model. This residual is then modeled with an SVR model at the next coarsest scale γ_1. A new residual is computed by taking the difference of the old residual and the γ_1 SVR model. This process is repeated until the pre-computed scales are exhausted.

The HSVR model contains a number of hyperparmeters that need to be specified, namely, ϵ, C_ℓ, the number of layers L to take, and the specific scales σ_ℓ for those layers. In [20], the authors chose to use exponential decay relationship between the scales, such as $\sigma_{\ell+1} = \sigma_\ell / \sqrt{decay}$. This, however, still leaves the critical choices of σ_0 and L unspecified. In all of our experiments that follow, we choose ϵ to be 1 percent of the variation of the signal

$$\epsilon = 0.01(\max f(x) - \min f(x)). \tag{21}$$

For each layer, C_ℓ was specified as

$$C_\ell = 5(\max_i r_{\ell-1}(x_i) - \min_i r_{\ell-1}(x_i)). \tag{22}$$

which was the choice given in [20]. Additionally, the *decay* variable was chosen to be 2 so that $\sigma_{\ell+1} = \sigma_\ell / \sqrt{decay}$. Python's scikit-learn library [21] was used throughout this paper. Its implementation of SVR requires the input parameter γ, rather than σ from (1). These two parameters are related as $\gamma = 1/\sigma^2$. Thus, the equivalent decay rate of the input parameter is

$$\gamma_{\ell+1} = \gamma_\ell * decay = \gamma_\ell * 2. \tag{23}$$

In the next sections, we take up the task of efficiently determining the hyperparameters L and σ_0 without the expensive step of performing grid search or training any models.

3. Predicting the Depth of Models

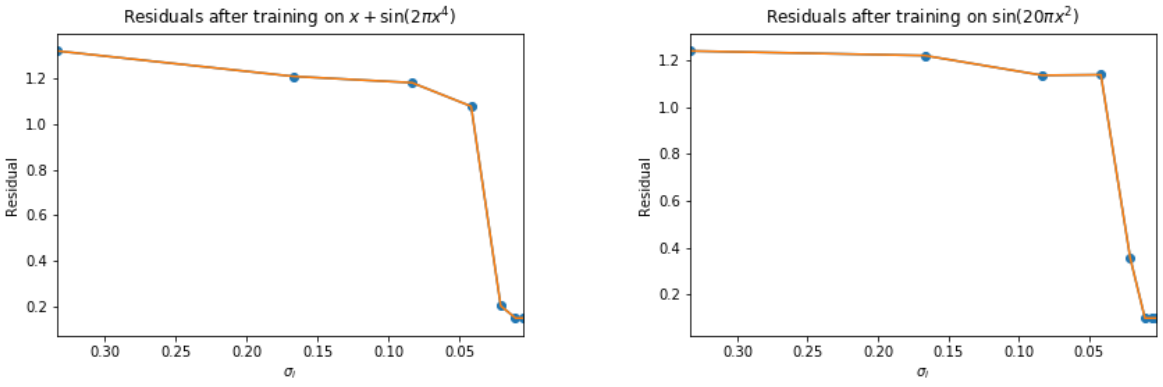
In this section, we will describe how error changes while training HSVR and we will provide methods of estimating the number of layers of such hierarchical model.

3.1. Phase Transition of the Training Error

For the rest of the paper, we have training data set $\{(x_{train}, y_{train})\}$ and testing data set $\{(x_{test}, y_{test})\}$ and the model $S(x) = \sum_{\ell=1}^{L} a_\ell(x; \sigma_l)$, as in (19). Let $S_i(x) = \sum_{\ell=1}^{i} a_\ell(x; \sigma_l)$ for $i = 1, \ldots, L$. For each layer, the HSVR (prediction) error is calculated as

$$r_i = \max_{\{(x_{test}, y_{test})\}} |y_{test} - S_i(x_{pred})| = \max_{\{(x_{test}, y_{test})\}} |y_{test} - y_{i,pred}|, \tag{24}$$

where we have denoted $y_{i,pred} = S_i(x_{pred})$. Therefore, r_i denotes the maximum error between the true signal at the test points x_{pred} and an HSVR model with i layers. By examining the change of the values r_i, it can be seen that there is a sudden drop in error, almost until tolerance ϵ, so that adding additional layers cannot reduce error any more. We will call that value the critical-sigma and denote it with σ_c. The drop in error can be seen in Figure 4.



(**a**) $f(x) = x + \sin(2\pi x^4)$ (**b**) $f(x) = \sin(20\pi x^2)$

Figure 4. Residuals while training HSVR model with decreasing σ as shown on x-axis. Both HSVR models exhibit a phase transition in their approximation error.

3.2. Critical Scales: Intuition and the Fourier Transform

Since the SVR models are fitting the data with Gaussians, a heuristic for choosing the scale will be shown on basic examples of periodic function and then expanded on more general cases in next sections.

Let $f(x) = \sin(2\pi f x)$, where x is in meters and f is the frequency in cycles per meter. The frequency will be related to the scale, σ, of the Gaussian:

$$G_\sigma(x, c) = \exp\left(-\frac{\|x - c\|_2^2}{\sigma^2}\right). \tag{25}$$

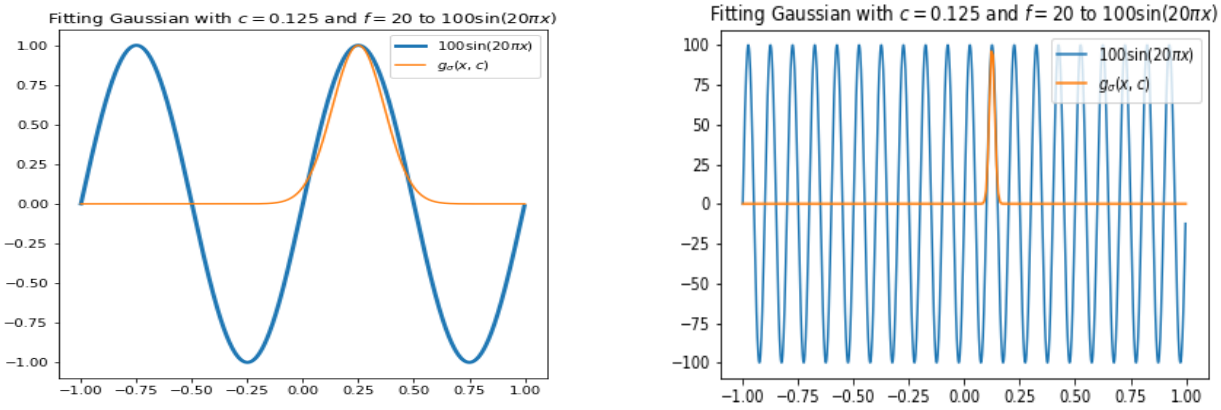
To relate the maximum frequency, f, to the scale, σ, we use the heuristic that we want 3 standard deviations of the gaussian to be half of the period. That is we want

$$3\sigma = \frac{T}{2}, \tag{26}$$

Since $T = \frac{1}{f}$, then

$$3\sigma = \frac{1}{2f}. \tag{27}$$

Figure 5 justifies this heuristic.



(**a**) Gaussian with mean 0.25 and standard deviation $\sigma = 1/6$ compared with $\sin(2\pi x)$

(**b**) 100 times a Gaussian with mean 0.125 and deviation $\sigma = 1/20$ compared with $100\sin(20\pi x)$

Figure 5. Fitting Gaussians to half-periods of sinusoids using the heuristic (27).

3.3. Determining Scales with FFT

We assume that we can learn HSVR model with scales σ corresponding to important frequencies in the FFT of the signal. The assumption is that these will give an information how to train the model. If in the FFT of the signal, there are a lot of frequencies, we assume that we need more HSVR layers and each will learn the most dominant scale at this level.

Our assumption is that required number of scales in the HSVR model can be determined by the data. If there are a lot of frequencies in FFT of the signal related to relatively big coefficients, we can assume that signal is more challenging for single SVR to model it. As in [20], we refine the scales. Here we use exponential decay and use only frequencies for which corresponding coefficients in FFT are large enough in order to avoid numerical problems and adding insignificant frequencies. The procedure is summarized in Algorithm 1. The *Filtering scales* algorithm is summarized in Algorithm 2. The model building is outlined in Algorithm 3.

Algorithm 1 Determining scales of HSVR model

Input: $(x_i, y_i), i = 0, \cdots n-1$, where x_i are equidistant points in domain and y_i values of function we want to model
1: $dx = x[1] - x[0]$
2: $freq =$ FFT frequencies of the signal
3: $C = FFT(y)$
4: $C = C/\max(|C|)$ # normalize coefficients respect to $L1$-norm
5: $freq_{support} = freq[|C| > 0.01]$
6: $scales = dx/(6 * freq_{support})$
7: $scales$ = sort scales in descending order
8: $scales = filter(scales)$
9: **return** $scales$

Algorithm 2 Filtering scales

Input:
$scales$ = vector of scales determined from FFT,
$decay$
1: $scales_{filtered} = [scales[0]]$
2: $n = len(scales)$
3: **for** i in range(1,n):
4: **if** $scales_{filtered}[-1]/scales[i] >= decay$:
5: $scales_{filtered}.append(scales[i])$
6: **return:** $scales_{filtered}$

Algorithm 3 Train HSVR

Input: $(x_i, y_i), i = 0, \cdots n-1$
scales (output of Algorithm 1)
1: $\epsilon = 0.01(\max_i(y_i) - \min_i(y_i))$
2: $r_0 = y = [y_0, \ldots, y_{n-1}]$
3: model = [ ] # comment: empty list to hold the SVR model at each layer
4: $m = len(scales)$ # comment: number of HSVR layers
5: **for** i in range(0, m):
6: σ_i = scales[i]
7: $C_i = 5(\max(r_i) - \min(r_i))$
8: svr_i = fitted SVR on (x, r_i) with parameters σ_i, C_i and tolerance ϵ
9: predictions = svr_i.predict(x)
10: $r_{i+1} = r_i - predictions$
11: model.append(svr_i)
12: **return:** model

3.4. Determining Scales with Dynamic Mode Decomposition

Let $(x_i, y_i), i = 0, \ldots, n$ be training set. Output data y_i is organized into Hankel matrix Y with M rows and N columns $(M > N)$. We will extract relevant frequencies with Hankel DMD, described in [22], using the DMD_RRR described in [23]. The DMD_RRR algorithm returns the residuals (rez) which determine how accurately the eigenvalues are computed, the eigenvalues (λ), and the eigenvectors (Vtn). As before, suppose we have a signal

$$f(x_n) = f(n\delta x), \quad (n = 0, \ldots, N-1). \tag{28}$$

We map this scalar-valued functions into a higher-dimensional space by delay-embedding. We choose $M < N$. The delay-embedding of the signal is the matrix

$$H = \begin{bmatrix} f(x_0) & f(x_1) & \dots & f(x_{N-M}) \\ f(x_1) & f(x_2) & \dots & f(x_{N-M-1}) \\ \vdots & \vdots & \ddots & \vdots \\ f(x_{M-1}) & f(x_M) & \dots & f(x_{N-1}) \end{bmatrix}, \tag{29}$$

so that for $j = 0, \dots, N - M$

$$H[:, j] = \begin{bmatrix} f(x_j) \\ f(x_{j+1}) \\ \vdots \\ f(x_{M+j-1}). \end{bmatrix} \tag{30}$$

We define a generalized Hankel matrix as an $m \times n$ rectangular matrix whose entries $H_{i,j}$ satisfy

$$H_{i,j} = H_{i+k,j-k} \tag{31}$$

for all indices such that $0 \leq i, i + k \leq m - 1$ and $0 \leq j, j - k \leq n - 1$, where $k \in \mathbb{Z}$. In simpler terms, a generalized Hankel matrix is a rectangular matrix that is constant on anti-diagonals. A generalized Hankel matrix can also be thought of a submatrix of a larger, regular Hankel matrix. Clearly, the delay-embedding of the scalar signal, Equaion (29), is an example of a generalized Hankel matrix which is why it was denoted as H.

The input matrices for DMD algorithms will be $\mathbf{X}$ and $\mathbf{Y}$, where $\mathbf{X}$ is the first $N - M$ columns of $\mathbf{H}$ and $\mathbf{Y}$ is the last $N - M$ columns. Frequencies are calculated as follows:

$$\omega_i = \frac{1}{2\pi i} \ln\left(\frac{\lambda_i}{|\lambda_i|}\right); \tag{32}$$

i.e., we just scale the DMD eigenvalues so that they are on the unit circle and then extract frequency of the resulting complex exponential, $\exp(i2\pi(f_\lambda \Delta x)) = \lambda/|\lambda|$. Let Ω contain the values $\omega_\lambda = f_\lambda \Delta x$. These are directly analogous to the frequencies computed using FFT.

We replace the support of the FFT with the support of the DMD frequency as follows. We will take all the values of Ω whose corresponding residual is less than some specified tolerance, `tol`, *and* whose corresponding mode's norm is greater than some percentage, η, of the total power of the modes. In other words, we only consider the frequencies which were calculated accurately enough and whose modes give a significant contribution to the signal; $\|mode[j]\|$ is analogous to the modulus of a FFT coefficient.

For each mode $Vtn[:, i]$ the energy is c_i for which $\|Y[:, 0] - c * Vtn[:, i]\|_2$ is minimal. Total power is defined as

$$T = \left(\sum_{v_i} c_i^2\right)^{\frac{1}{2}}. \tag{33}$$

Like in determining σ_c using FFT, the frequency support of S_{DMD} is defined as

$$S_{DMD} = \{\omega_i : rez[i] < tol, |c_i| > \eta T\}, \tag{34}$$

where $rez[i]$ is residual corresponding to the i-th mode. These considerations can be summarized in Algorithm 4.

Algorithm 4 Estimating scales from data using Hankel DMD

Input:
time step: Δx,
time series f: $f[n] = f(\Delta x n)$,
length of time series vector: N,
tolerance for support: tol, η, M: number of rows of Hankel matrix
1: H = Hankel matrix made from f with M rows and N-M columns
2: $rez, \lambda, Vtn = DMD\_RRR(H)$
3: $\omega = \frac{1}{2\pi i} \ln\left(\frac{\lambda}{|\lambda|}\right)$
4: $T = 0$
5: **for** i = 0 to N - 1:
6: $\quad E[i] = |\langle Y[:,0], Vtn[:,i]\rangle|$
7: $\quad T = T + E[i]^2$
8: $T = \sqrt{T}$
9: $S_{DMD} = []$
10: **for** i = 0 to N-1:
11: $\quad$ **if** $rez[i] < tol$ and $energy[i] > \eta T$
12: $\quad\quad S_{DMD}.append(\omega[i])$
13: **return** $\sigma_{DMD} = \frac{\Delta x}{6 S_{DMD}}$

4. Results

In this section, results of our methods are provided. We demonstrate our methods on explicitly defined function, system of ODEs, and finally on vorticity data from a fluid mechanics simulations. In all cases,

$$\epsilon = 0.01\left(\max_{\{y_{train}\}} y_{train} - \min_{\{y_{train}\}} y_{train}\right) \tag{35}$$

and the error is calculated as in (24).

4.1. Explicitly Defined Functions

We model explicitly defined functions $f(x)$ on $[0, 2]$. The dataset consists of 2001 equidistant points in that interval, where every other point is used for the training set. The error is calculated as the maximum absolute value of difference between prediction and actual value (Equation (24)). Results are summarized in Table 1. The parameter ϵ is specified as above. Each method predicts a certain number of layers required to push the model error close to ϵ. As seen in the table, both the FFT and DMD approaches produce models that give similar errors (near ϵ). Although the DMD approach results in models with less layers in general, it fails for polynomials and the exponential function. This is because we normalize eigenvalues to the unit circle. An extension of this could use the modulus $|\lambda|$ as well as ω in $e^{\lambda + i\omega}$.

4.2. ODE's

We will demonstrate the methods on modeling solutions of the Lorenz system of ODE

$$x(t) = -10x + 10y, \tag{36}$$
$$y(t) = 28x - y - xz, \tag{37}$$
$$z(t) = xy - \frac{8}{3}z, \tag{38}$$

with initial conditions $x_1(0) = 1.0$, $x_2(t) = 1.0$, $x_3(t) = 1.0$, on $[0, 10]$. The system evolved solved for 500 equidistant time steps which was used for training set. A solution consisting of 2000 equidistant time steps is then used for test set. Each $x(t)$, $y(t)$, $z(t)$ were regarded as separate signals to model. All hyperparameters are determined as before. Results are

summarized in Table 2. For both systems, we can see that error drops nearly to ϵ, but the DMD approach results in models with less layers.

Table 1. Results for explicitly defined functions, when using scales determined from FFT with decay 2 and ϵ given by (35), for e^x DMD did not output frequencies different from 0 (entries denoted by $*$ in the corresponding row).

Function	ϵ	Predicted # of Layers (FFT)	Error (FFT)	Predicted # of Layers (DMD)	Error (DMD)
$\sin(2\pi x)$	0.02	1	0.02	1	0.02
$\sin(20\pi x)$	0.0199	1	0.021	1	0.02
$\sin(200\pi x)$	0.019	1	0.093	1	0.097
$100\sin(20\pi x)$	1.99	1	2	1	2.01
$40\cos(2\pi x)$	0.8	1	0.8	1	0.8
$100\cos(20\pi x)$	2	1	2.03	1	2
$\sin(2\pi x^2)$	0.0199	5	0.02	1	0.02
$x + x^2 + x^3$	0.14	2	0.14	1	8
e^x	0.063	1	0.064	*	*
$x + \sin(2\pi x^4)$	0.03	7	0.037	1	0.034
$\cos(2\pi x) + \sin(20\pi x)$	0.0397	2	0.0404	2	0.042
$\cos(20\pi x)\sin(15\pi x)$	0.02	2	0.021	2	0.022
$\cos(32\pi x)^3$	0.0199	1	0.022	2	0.022
$\sin(13\pi x) + \sin(17\pi x) + \sin(19\pi x) + \sin(23\pi x)$	0.076	1	0.077	1	0.077
$\sin(50\pi x)\sin(20\pi x)\cos(15\pi x)$	0.0187	3	0.02	2	0.02
$\sin(40\pi x)\cos(10\pi x) + 3\sin(20x)\sin(40x)$	0.064	5	0.065	3	0.066
$\sin(2x)\cos(32x)$	0.0198	5	0.02	1	0.02

Table 2. Results of training HSVR with scales used from FFT and DMD on Lorenz system.

Function	ϵ	Predicted # of Layers (FFT)	Error (FFT)	Predicted # of Layers (DMD)	Error (DMD)
x(t)	0.314	6	0.325	2	0.324
y(t)	0.408	6	0.469	2	0.469
z(t)	0.468	5	0.485	2	0.494

4.3. Vorticity Data

This data was provided by Georgia Institute of Technology [24,25]. It contains information about vorticity of a fluid field is some computational box. For each point in space, we want to model vorticity at that point as time evolves. Vorticity in every other time step is used for training and we test our models on rest of the time steps. We analyzed doubly periodic and non periodic data.

4.3.1. Doubly Periodic Data

This data was generated assuming doubly periodic boundary conditions. The dataset contains information about the vorticity on 128×128 equidistant space-grid on $[0, 0.1016]^2$. There are in total 1201 snapshots with time step $dt = 0.03125$. This results in a tensor of dimensions $128 \times 128 \times 1201$. For each fixed point in space, there is a signal with 1201 time steps. A few examples of such signals are shown in Figure 6.

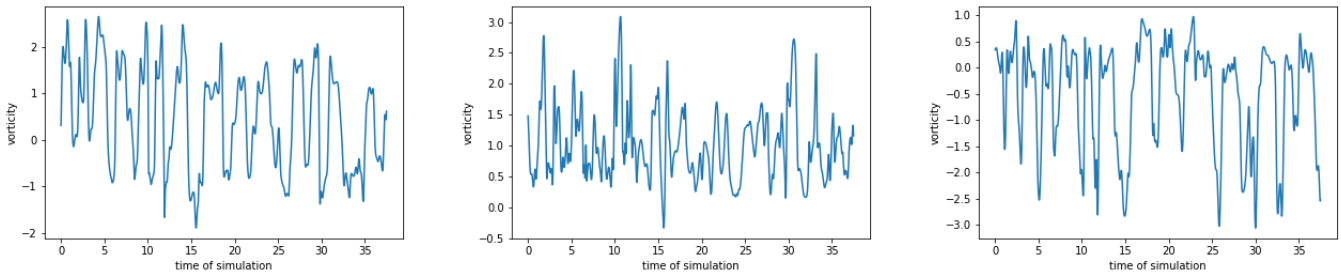


Figure 6. Examples of vorticity at 3 different space-points for fluid simulations with doubly periodic boundary conditions.

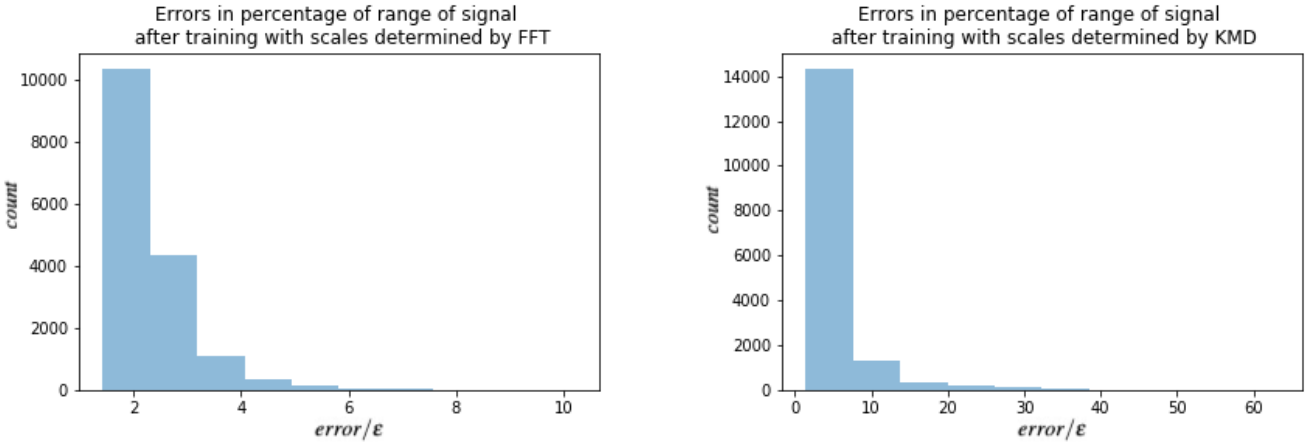
For each of these signals, every other point in time is used for training the HSVR model, which results in total of 128×128 models. Results are summarized in Table 3. For each model, the error is calculated as in (24) with $i = L$, where L is number of layers. Since ϵ is given by (35), i.e., 1% of the range of the training data, the ratio error$/\epsilon$ gives a measure of error in percentages of range of signal. A ratio of 2 would imply that the maximum error of the model over the test set was only 2% of the range of the test data; i.e.,

$$2 = \frac{error}{\epsilon} \iff error = 2\epsilon = 0.02\left(\max_{\{y_{train}\}} y_{train} - \min_{\{y_{train}\}} y_{train}\right). \tag{39}$$

Histograms of these ratios for both FFT and DMD are shown in Figure 7. Most models produced on error close the ϵ threshold (a perfect match would give a ratio of 1).

Table 3. Results after training HSVR on doubly periodic vorticity data.

	ϵ	Predicted # of Layers (FFT)	Error (FFT)	Predicted # of Layers (DMD)	Error (DMD)
min	0.0199	6	0.038	1	0.0399
mean	0.0354	8	0.082	3	0.148
max	0.0488	9	0.284	5	2.463



(**a**) Scales determined by FFT

(**b**) Scales determined by DMD

Figure 7. Histograms of $error/\epsilon$ for models trained on vorticity data with doubly periodic boundary conditions. *error* is the model error given by (24) (with $i = L$) and ϵ is given by (35). There were $128^2 = 16{,}384$ total models trained. The count on the vertical axis is the number of models that fell into the corresponding bin.

4.3.2. Non Periodic Data

The data contains information about vorticity on 359×279 equidistant space-grid on with step $dx = dy = 0.05$. No assumption of periodicity of boundary condition was made. There are in total 1000 snapshots with time step $dt = 1$ ms. This results in tensor $359 \times 279 \times 1000$. Similar to the dataset with periodic boundary conditions, the HSVR model is trained for each fixed point in space, which results in 359×279 models. The error's and ϵ's are calculated as before. A few examples of such signals are in Figure 8. Results are summarized in Table 4 and a histogram of ratios error$/\epsilon$ for FFT and DMD are shown in Figure 9. For both doubly periodic and non periodic data, we can see that a large majority of the models have a ratio between 1 and 2 which means the maximum error of a majority models is $error \leq 2\epsilon = 0.02(\max_{\{y_{train}\}} y_{train} - \min_{\{y_{train}\}} y_{train})$ which is less than 2% of the range of the training data. For DMD approach, the majority of models have ratios less 10 which corresponds to a maximum error of $error \leq 10\epsilon = 0.01(\max_{\{y_{train}\}} y_{train} - \min_{\{y_{train}\}} y_{train})$, which is less than 10% of the range of the training data. From Tables 3 and 4 we can see that DMD estimates less layers, but with bigger error after training.

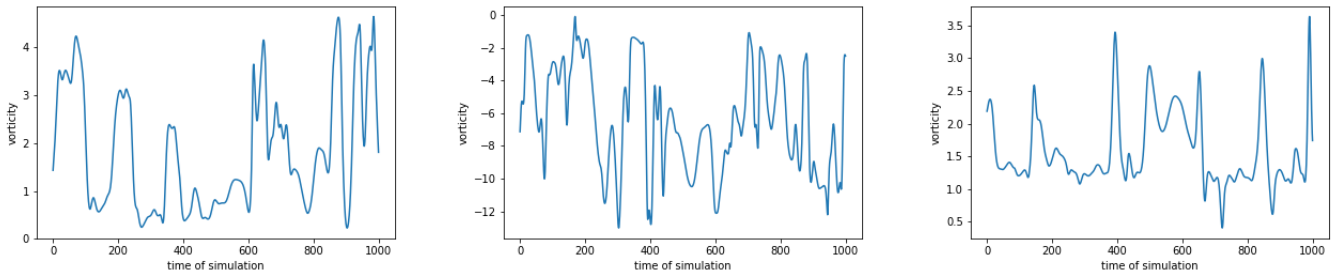
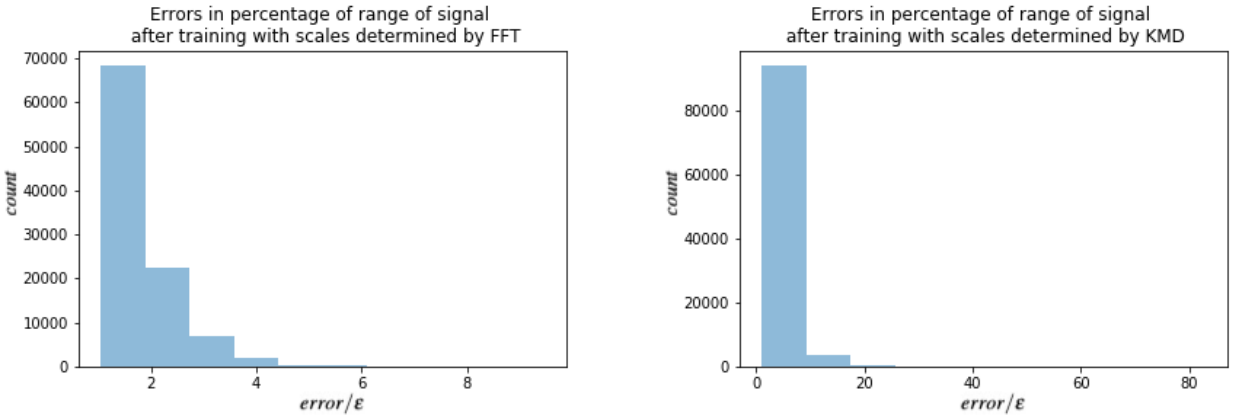


Figure 8. Examples of vorticity in 3 space-points for non periodic data.

Table 4. Results after training HSVR on non periodic vorticity data.

	ϵ	Predicted # of Layers (FFT)	Error (FFT)	Predicted # of Layers (DMD)	Error (DMD)
min	0.019	5	0.0006	2	0.0006
mean	0.035	7	0.0975	3	0.197
max	0.048	9	0.667	7	6.137



(**a**) Scales determined by FFT (**b**) Scales determined by DMD

Figure 9. Histograms of $error/\epsilon$ for models trained on vorticity data with non-periodic boundary conditions. *error* is the model error given by (24) (with $i = L$) and ϵ is given by (35). There were $359 \times 279 = 100{,}161$ total models trained. The count on the vertical axis is the number of models that fell into the corresponding bin.

5. Discussion

Both approaches (FFT and DMD) to estimating the number of layers required by HSVR to push the modeling error close to the ϵ threshold show promise and allow computation of the HSVR depth a priori. While the DMD approach often predicted a smaller number of layers, usually with comparable error, our choice of unit circle normalization prevented it from performing well on functions without oscillation. Furthermore, the DMD algorithm itself comes from the dynamical systems community and requires that the domain of the signal (the x variables) be strictly ordered. For the explicitly defined functions we considered, there was a strict spatial ordering since the domains of the functions were intervals on the real line. For the vorticity data, the time signals at each spatial point were to be modeled and therefore the data points could be strictly ordered in time. For functions whose domain is multi-dimensional, say $\mathbb{R}^2$, there is no strict ordering and the DMD approach, as formulated here, would break down. The method based on the Fourier transform seems to have more promise in analyzing multivariable, multiscale signals, as it can compute multidimensional wave vectors.

A recent paper [26] also deals with the number of layers of a model and "scales". However, in the mentioned paper, the authors are concerned with how far a signal or gradients will propagate through a network before dying. The scales they compute control how many layers the gradient or signal can propagate before they die. If the network is too deep the gradients go to zero before fully backpropagating through the network, resulting in an untrainable network. The result they compute is a fundamental characteristic of the network and is independent of the dataset. It does not matter what the input data is: constant, single scale, multiscale, etc, the scale parameters they compute are not affected.

Conversely, we are most focused on multiscale signals and tailoring the architecture to best represent such signals. The scales we compute are inherent properties of the dataset, not inherent properties of the network model. The length of the network adapts to the scales contained in the dataset.

It would be interesting in the future to combine the two methodologies, with the methods in [26] giving bounds on the number of layers and then adapting our techniques to analyze the inherent scales in the dataset. This could tell us whether a simple, fully, connected, feedforward network could adequately represent the signal or if something more complex was needed.

6. Conclusions

In this work we presented method for choosing hyperparameters for HSVR from time-series data only. We described two approaches, using FFT or DMD. We saw that estimating hyperaparameters with FFT results in a model with more layers and smaller error, whereas the DMD approach gave models that had less layers (and thus more efficient models).

Author Contributions: Conceptualization, R.M., M.F., Z.D. and I.M. (Igor Mezić); Data curation, I.M. (Iva Manojlović); Validation, I.M. (Iva Manojlović); Writing—original draft, I.M. (Iva Manojlović); Writing—review & editing, R.M., M.F., Z.D. and I.M. (Igor Mezić). All authors have read and agreed to the published version of the manuscript.

Funding: This research was partially supported by the DARPA contract HR0011-18-9-0033, the Air Force Office of Scientific Research contract FA9550-17-C-0012, and the Croatian Science Foundation grant IP-2019-04-6268.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not available.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Joachims, T. The Maximum-Margin Approach to Learning Text Classifiers: Methods Theory, and Algorithms. In *Ausgezeichnete Informatikdissertationen*; Lecture Notes in Informatics (LNI); Koellen Verlag: Bonn, Germany, 2002. Available Online: https://dl.gi.de/bitstream/handle/20.500.12116/4447/GI-Dissertations.02-6.pdf?sequence=1 (accessed on 24 December 2020).
2. Wahba, G. Support vector machines, reproducing kernel Hilbert spaces and the randomized GACV. *Adv. Kernel-Methods-Support Vector Learn.* **1999**, *6*, 69–87.
3. Vapnik, V.; Chapelle, O. Bounds on error expectation for support vector machines. *Neural Comput.* **2000**, *12*, 2013–2036. [CrossRef]
4. Ong, C.S.; Smola, A.J.; Williamson, R.C. Learning the kernel with hyperkernels. *J. Mach. Learn. Res.* **2005**, *6*, 1043–1071.
5. Hutter, F.; Kotthoff, L.; Vanschoren, J. (Eds.) Hyperparameter Optimization. In *Automated Machine Learning: Methods, Systems, Challenges*; Springer International Publishing: Cham, Switzerland, 2019; pp. 3–33. [CrossRef]
6. Yu, T.; Zhu, H. Hyper-Parameter Optimization: A Review of Algorithms and Applications. *arXiv* **2020**, arXiv:2003.05689.
7. Chapelle, O.; Vapnik, V.; Bousquet, O.; Mukherjee, S. Choosing multiple parameters for support vector machines. *Mach. Learn.* **2002**, *46*, 131–159. [CrossRef]
8. Chung, K.M.; Kao, W.C.; Sun, C.L.; Wang, L.L.; Lin, C.J. Radius margin bounds for support vector machines with the RBF kernel. *Neural Comput.* **2003**, *15*, 2643–2681. [CrossRef]
9. Gold, C.; Sollich, P. Model selection for support vector machine classification. *Neurocomputing* **2003**, *55*, 221–249. [CrossRef]
10. Hooke, R.; Jeeves, T.A. "Direct Search" Solution of Numerical and Statistical Problems. *JACM* **1961**, *8*, 212–229. [CrossRef]
11. Kirkpatrick, S.; Gelatt, C.D.; Vecchi, M.P. Optimization by simulated annealing. *Science* **1983**, *220*, 671–680. [CrossRef]
12. Mallick, B.K.; Ghosh, D.; Ghosh, M. Bayesian classification of tumours by using gene expression data. *J. R. Stat. Soc. Ser. B (Stat. Methodol.)* **2005**, *67*, 219–234. [CrossRef]
13. Friedrichs, F.; Igel, C. Evolutionary tuning of multiple SVM parameters. *Neurocomputing* **2005**, *64*, 107–117. [CrossRef]
14. Frohlich, H.; Chapelle, O.; Scholkopf, B. Feature selection for support vector machines by means of genetic algorithm. In Proceedings of the 15th IEEE International Conference on Tools with Artificial Intelligence, Sacramento, CA, USA, 5 November 2003; pp. 142–148.
15. Igel, C. Multi-objective model selection for support vector machines. In Proceedings of the International Conference on Evolutionary Multi-Criterion Optimization, Guanajuato, Mexico, 9–11 March 2005; Springer: Berlin/Heidelberg, Germany, 2005; pp. 534–546.
16. Cortes, C.; Vapnik, V. Support-vector networks. *Mach. Learn.* **1995**, *20*, 273–297. [CrossRef]
17. Smola, A.J.; Schölkopf, B. A tutorial on support vector regression. *Stat. Comput.* **2004**, *14*, 199–222. [CrossRef]
18. Schmid, P.J. Dynamic mode decomposition of numerical and experimental data. *J. Fluid Mech.* **2010**, *656*, 5–28. [CrossRef]
19. Rowley, C.W.; Mezić, I.; Bagheri, S.; Schlatter, P.; Henningson, D.S. Spectral analysis of nonlinear flows. *J. Fluid Mech.* **2009**, *641*, 115–127. [CrossRef]
20. Bellocchio, F.; Ferrari, S.; Piuri, V.; Borghese, N.A. Hierarchical approach for multiscale support vector regression. *IEEE Trans. Neural Netw. Learn. Syst.* **2012**, *23*, 1448–1460. [CrossRef]
21. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
22. Arbabi, H.; Mezić, I. Ergodic theory, dynamic mode decomposition, and computation of spectral properties of the Koopman operator. *SIAM J. Appl. Dyn. Syst.* **2017**, *16*, 2096–2126. [CrossRef]
23. Drmač, Z.; Mezić, I.; Mohr, R. Data driven modal decompositions: analysis and enhancements. *SIAM J. Sci. Comput.* **2018**, *40*, A2253–A2285. [CrossRef]
24. Tithof, J.; Suri, B.; Pallantla, R.K.; Grigoriev, R.O.; Schatz, M.F. Bifurcations in quasi-two-dimensional Kolmogorov-like flow. *J. Fluid Mech.* **2017**, 837–866. [CrossRef]
25. Suri, B.; Tithof, J.; Mitchell, R.; Grigoriev, R.O.; Schatz, M.F. Velocity profile in a two-layer Kolmogorov-like flow. *Phys. Fluids* **2014**, *26*, 053601. [CrossRef]
26. Schoenholz, S.S.; Gilmer, J.; Ganguli, S.; Sohl-Dickstein, J. Deep Information Propagation. *arXiv* **2017**, arXiv:1611.01232.