Fast variable selection makes scalable Gaussian process BSS-ANOVA a speedy and accurate choice for tabular and time series regression

Anonymous Author(s)
Affiliation
Address
email

Abstract

Gaussian processes (GPs) are non-parametric regression engines with a long history. They are often overlooked in modern machine learning contexts because of scalability issues: regression for traditional GP kernels are $\mathcal{O}(N^3)$ where N is the size of the dataset. One of a number of scalable GP approaches is the Karhunen-Loéve (KL) decomposed kernel BSS-ANOVA, developed in 2009. It is $\mathcal{O}(NP)$ in training and $\mathcal{O}(P)$ per point in prediction, where P is the number of terms in the ANOVA / KL expansion. A new method of forward variable selection, enabled by the ordered nature of the basis functions and fast Gibbs sampling, quickly and effectively limits the number of terms, yielding a method with competitive accuracies, training and inference times for large tabular datasets. The new algorithm determines how high the orders of included terms should reach, balancing model fidelity with model complexity using Bayesian and Akaike information criteria (BIC/AIC). The inference speed and accuracy makes the method especially useful for modeling dynamic systems in a model-free manner, by modeling the derivative in a dynamic system as a static problem, then integrating the learned dynamics using a high-order scheme. The methods are demonstrated on a 'Susceptible, Infected, Recovered' (SIR) toy problem, with the transmissibility used as forcing function, along with the 'Cascaded Tanks' benchmark dataset. Comparisons on the static prediction of derivatives are made with a Random Forest and Residual Neural Network, while for the timeseries prediction comparisons are made with LSTM and GRU recurrent neural networks. The GP outperforms the other methods in all modeling tasks on accuracy, while (in the case of the neural networks) performing many orders of magnitude fewer calculations. For the SIR test, which involved prediction for a set of forcing functions qualitatively different from those appearing in the training set, the GP captured the correct dynamics while the neural networks failed to do so.

1 BSS-ANOVA Gaussian processes

1.1 Gaussian process fundamentals

Gaussian processes (GPs) are stochastic functions that are engines for nonparametric regression. Initially developed for modeling and interpolation in geographic information systems datasets, applications have multiplied across many fields of data science, including and especially model-aided interpretation of data[Kennedy and O'Hagan, 2001]. A key advantage of the GP is its broad, continuous nonparametric support and the frequent amenability of different GP kernels to precise analysis.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

A GP is Gaussian in that it is a covariance model linking pairs of points on functional draws. As such a GP is completely described by a mean function (often zero in the prior) and covariance kernel. The most famous and perhaps simplest of the covariance kernels is the squared exponential:

$$\Gamma(\vartheta, \vartheta') = \varsigma^2 \exp\left[\frac{(\vartheta - \vartheta')^2}{\xi}\right] \tag{1}$$

where the sill ς^2 and range ξ parameters determine the scale and smoothness of the draws. In a typical implementation modeling a static dataset Z, the statistical model

$$Z = \delta(\vartheta|\varsigma^2, \xi) + \epsilon \tag{2}$$

with ϵ an observation error process, is first used to infer the hyperparameters, after which predictions conditioned on the training dataset can be made. The draws on the squared exponential GP – a limiting case of the Matérn covariance family – are infinitely differentiable in principle.

1.2 Scalable Gaussian processes and BSS-ANOVA

From a practical standpoint the training of the above GP is $\mathcal{O}(N^3)$, requiring a Cholesky decomposition of the full covariance matrix. This limits the use of the GP to moderately-sized datasets, generally of a thousand instances or fewer.

Scalable GPs aim to overcome this difficulty through a number of strategies. Liu et al. [2020] provide a thorough overview of efforts that aim to improve scalability while maintaining prediction accuracy. Many techniques use global kernel approximations derived in some sense from a set of $M << N$ inducing points [Chalupka et al., 2013, Quinonero-Candela and Rasmussen, 2005, Deisenroth and Ng, 2015, Rasmussen and Ghahramani, 2001, Wang et al., 2022] where the goal is to approximate the full-rank kernel matrix with local approximations. Of particular note is a $\mathcal{O}(N)$ method that directly estimates the covariance with training and inference times that limits the increase in M for large N developed by Wilson et al. [2015].

In a distinct line of research from the inducing points-based developments in scalable GPs, Reich et al. [2009] introduced the decomposable Bayesian Smoothing Spline ANOVA (BSS-ANOVA) kernel, which is subject to first an ANOVA decomposition among the inputs (main effects, two-way interactions, etc.) followed by a Karhunen-Loéve (KL) decomposition. This yields a set of spectral, nonparametric, orthogonal, ordered and – crucially – *deterministic* basis functions that are precomputable when inputs are normalized on a [0,1] interval. The (non-stationary) kernel for main effects is:

$$\Gamma_1(\vartheta, \vartheta') = \mathcal{B}_1(\vartheta)\mathcal{B}_1(\vartheta') + \mathcal{B}_2(\vartheta)\mathcal{B}_2(\vartheta') + \frac{1}{24}\mathcal{B}_4(|\vartheta - \vartheta'|) \tag{3}$$

where $\mathcal{B}_k$ is the kth Bernoulli polynomial. This kernel is effectively a sum of a non-stationary quadratic response surface (corresponding to the first two terms) and a stationary deviation (the final term). Covariances for higher-order interactions are constructed with dyadic products of the main effect covariance:

$$\Gamma_2([\vartheta_j, \vartheta_k], [\vartheta'_j, \vartheta'_k]) = \Gamma_1(\vartheta_j, \vartheta'_j)\Gamma_1(\vartheta_k, \vartheta'_k) \tag{4}$$

such that the full kernel for a system with n inputs is written $\delta \sim MVN(0, \Gamma)$, with

$$\Gamma = \sigma_0^2\tau_0^2 + \sigma_1^2\tau_1^2 \sum_{i=1}^{n} \Gamma_{1,i} + \sigma_2^2\tau_2^2 \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \Gamma_{2,ij} + \cdots \tag{5}$$

The kernel so constructed is supported by a second-order Sobolev space [Reich et al., 2009], which is a very broad and dense class of functions. The KL decomposition of 5 yields an expansion

$$\delta(\vartheta; \beta) = \beta_0 + \sum_{i=1}^{n}\sum_{k=1}^{\infty} \beta_{ik}\phi_k(\vartheta_i) + \sum_{i=1}^{n-1}\sum_{j=i+1}^{n-1}\sum_{k=1}^{\infty}\sum_{l=1}^{\infty} \beta_{ik,jl}\phi_k(\vartheta_i)\phi_l(\vartheta_j) + \cdots \tag{6}$$

with ϕ the deterministic basis functions, precomputable if inputs are normalized to a [0,1] interval. Given a statistical model

$$z_k = \delta_k(\{\vartheta_i\}_k; \beta) + \epsilon \tag{7}$$

with ϵ a white noise observation error, and an appropriate cutoff to the KL expansion 6, the model is linear in the coefficients β.

1.3 Gibbs sampling

Given the assumption

$$\sigma_0^2\tau_0^2 = \sigma_1^2\tau_1^2 = \sigma_2^2\tau_2^2 = \cdots = \sigma^2\tau^2 \tag{8}$$

then the priors for the coefficients are independent normal

$$\beta_{\cdot k} \sim N(0, \lambda_k\sigma^2\tau^2)\,,\ \beta_{\cdot k,\cdot l} \sim N(0, \lambda_k\lambda_l\sigma^2\tau^2)\,,\ \cdots \tag{9}$$

where λ_k is the eigenvalue for the eigenfunction ϕ_k. For convenience we multiply each basis function by its eigenvalue such that $\beta \overset{\text{iid}}{\sim} N(0, \tau^2)$. (Basis functions are then evaluated on a dense grid of 500 intervals, and fit to cubic splines.) If the variance of the white noise observation error is σ^2, with inverse gamma prior $\sigma^2 \sim IG(a, b)$, with a and b the shape and scale parameters, respectively; and if τ^2 has inverse gamma prior $\tau^2 \sim IG(a_\tau, b_\tau)$, then an iterative Gibbs sampler can be devised such that for fixed $\{\sigma^2, \tau^2\}$, $\beta \sim MVN(\mu, \Sigma)$, with

$$\mu = (X^TX + 1/\tau^2)^{-1}X^TZ \tag{10}$$

$$\Sigma = \sigma^2\Big(X^TX + 1/\tau^2 I\Big)^{-1} \tag{11}$$

where $X \in \mathbb{R}^{N\times P}$ is a matrix constructed from the basis functions, whose rows correspond to instances and columns to terms in the expansion. For fixed $\{\beta, \tau^2\}$, $\sigma^2 \sim IG(a^*, b^*)$, with

$$a^* = a + 1 + N/2 + P/2 \tag{12}$$

$$b^* = b + \frac{1}{2}\Big[(\mu-\beta)^T(X^TX + 1/\tau^2 I)(\mu-\beta) + Z^TZ - \mu^TX^TZ\Big] \tag{13}$$

For fixed $\{\beta, \sigma^2\}$, $\tau^2 \sim IG(a_\tau^*, b_\tau^*)$, with

$$a_\tau^* = a_\tau + (P-1)/2 \tag{14}$$

$$b_\tau^* = b_\tau + \frac{1}{2\sigma^2}\beta^T\beta \tag{15}$$

This algorithm is implemented in the routine 'gibbs_Xin' in the supplement.

2 Variable selection

It's clear from 6 that the number of terms in the expansion can increase rapidly, even for low-dimensional input spaces. A key component of applying the GP to a modeling problem is the selection of terms.

2.1 Indicator variable methods

Reich, Storlie and Bondell took a hierarchical approach to the problem, estimating a separate variance τ^2 for each term in the expansion, which is in turn expressed in terms of an indicator variable with a Bernoulli prior [Reich et al., 2009]. This approach, like other 'indicator variable' methods, accomplishes the variable selection and the training simultaneously and comprehensively, at the cost of requiring a large number of variables in the prior model and a Markov chain Monte Carlo (MCMC) sampling procedure. MCMC is significantly slower (and more difficult to implement) than Gibbs sampling.

2.2 Forward variable selection

Our approach takes advantage of the ordered nature of the basis functions, using forward variable selection. The algorithm proposes a series of models, using the Bayesian or Akaike Information Criteria (BIC/AIC) to determine when to stop the iterations.

The algorithm constructs models with terms having up to (optionally) three-way interactions (that is, terms depending on three different inputs at once). Terms are added in stages labeled by an integer

"index" that initializes at 1. At each stage, a series of substages cycle through all permutations of basis function orders that sum up to that stage's index. Stage 1 adds only first order main effects. Stage 2 adds second order main effects and first order two way interactions – corresponding to $\phi_1(\vartheta_i)\phi_1(\vartheta_j)$ – in two separate substages. The substages always occur such that terms involving lower-order basis functions (in the case of stage 2, this is the two-way interactions) come first. Each substage adds at once all combinations of inputs and all permutations among each combination. Then the sampler is called and the BIC or AIC is calculated. Because there is not a monotonic decrease / increase pattern for the discriminator, a "tolerance" setting controls how many substages the algorithm can iterate through without finding a new minimum BIC or AIC before it terminates. The algorithm returns the optimum model.

Algorithm 1 BSS-ANOVA forward variable selection algorithm

procedure FWDVARSELECT(ϑ, Z, ϕ, tol, h) ▷ h is a vector of hyperparameters.
  ind = 1
  count = 0
  while count < tol **do**
    if ind is new **then**
      Find all combinations of integers that sum up to ind, ordering them by the maximum integer appearing in each combination, with the lowest maximum first.
    1. Select the next combination in the set and place the integers into a vector with as many elements as there are model inputs, buffering out with zeroes.
    2. Produce a matrix M_d the rows of which contain all permutations of that vector.
    ▷ Each row corresponds to a term in the GP expansion.
    3. Produce an input matrix X_d where columns are model terms and rows are experiments, for all terms appearing in M_d.
    4. Recursively concatenate: $X = [XX_d]$, $M = [M; M_d]$
    5. β, BIC = gibbs_Xin(X, Z, ϕ, h)
    if the BIC is a minimum for all models **then**
      save the model
      count = 0
    else
      count++
    if all combinations for ind have been utilized **then**
      ind++
  Return M, β, BIC

This algorithm appears in the routine 'emulator_Xin' in the supplement.

3 Experiments and results: dynamic system identification

3.1 Procedure

BSS-ANOVA regression – as is the case for other GPs – is most effective for tabular datasets with continuous inputs and targets of moderate dimensionality. This suggests an application in dynamic system identification. Indeed BSS-ANOVA GPs have been utilized as components of other models ("intrusively") for this purpose in a number of applications [Bhat et al., 2017, Lei et al., 2019, Ostace et al., 2022]. We demonstrate here that they may also be used directly to identify dynamics in more general cases, without the aid of an accompanying model.

The procedure is a concurrent one, in that derivatives estimated from the datasets are modeled directly using BSS-ANOVA with forward variable selection, using the concurrent values of the system states and other inputs; for example a two-state system is modeled using two separate GPs:

$$\dot{x}_1 = \delta_1(x_1, x_2, u) \tag{16}$$

$$\dot{x}_2 = \delta_2(x_1, x_2, u) \tag{17}$$

The identified system is then integrated to yield predictions with uncertainty.

The procedure was demonstrated on two nonlinear dynamic datasets: a synthetic dataset derived from the susceptible, infected, recovered model (SIR model) for infectious disease, and the 'Cascaded

Table 1: Cascaded tanks 5-fold cross validated accuracies: derivatives

Method	$\dot{h}_1$ (MAE/10^{-4})	$\dot{h}_2$ (MAE/10^{-4})
BSS-ANOVA	18±6.5	39±3.6
RNN	36±14	61±15
RF	30±9.4	49±4.9

Table 2: Cascaded tanks 5-fold cross validated accuracies: timeseries

Method	h_1 (MAE/MAPE)	h_2 (MAE/MAPE)
BSS-ANOVA	0.1167±0.0382 / 4.67±1.58	0.1577±0.0334 / 5.99±1.75
LSTM	0.2345±0.1006 / 9.46±4.87	0.2296±0.0378 / 9.58±3.32
GRU	0.3243±0.1092 / 12.16±5.02	0.2481±0.0402 / 9.89±3.40

Tanks' experimental benchmark dataset. In both cases comparisons were made to long short term memory (LSTM) and gated recurrent unit (GRU) neural netowrks for timeseries prediction. In the case of the cascaded tanks benchmark comparisons were made against random forest (RF) and a residual neural network (RNN) for the static derivative estimation problem.

3.2 Benchmark dataset: cascaded tanks

The cascaded tanks nonlinear benchmark dataset is an experimental nonlinear dynamic system [Wigren and Schoukens, 2013]. The experiment consists of a set of two tanks and a reservoir of water. An upper tank is filled by a pump from the reservoir. An outlet in the upper tank empties into the lower tank, which in turn empties through an outlet back into the reservoir. A signal sent to the pump serves as the forcing function for the system, with the tank water level heights the two states of the system. A newer version of the dataset includes an 'overflow' phenomenon which is considered to be a 'hard nonlinearity' [Schoukens and Noel, 2017], however the newer dataset provides only a timeseries for the upper tank h_1 along with the pump signal u. We chose to demonstrate the novel GP approach and compare to neural network-based methods on the multivariate dataset.

We first compared the performance of the GP with RF and RNN static regressors. Derivatives were calculated via direct finite differences for the relatively noise-free dataset, yielding 10000 instances. Each method was trained on concurrent values of both states and the forcing function for each derivative. For the GP we used hyperparameters of $a = 1000$, $b = 1.001$, $a_\tau = 4$ and $b_\tau = 55$ for $\dot{h}_1$ and 69.1 for $\dot{h}_2$, with tolerances of 3 for $\dot{h}_1$ and 5 for $\dot{h}_2$, and the AIC as discriminator. Of 2000 draws the first 1000 were discarded. Only two-way interactions were required. For the RF 100 trees were used with a leaf size of 5. The RNN had a depth of 6 (filter sizes ranging from 16 to 64) and in between each fully connected layer is a batch normalization and relu layer. The mini batch size is 16, initial learn rate is 0.001, the data was shuffled every epoch for a total of 30 epochs, and the validation frequency was 1000. The 5-fold cross-validated results appear in Table 1. The GP performed best for both outputs by clear margins, followed by the RF and then the RNN.

Timeseries predictions follow for the GP via a 4th-order Runge-Kutta integration routine. These were compared with LSTM and GRU recurrent neural networks. For the LSTM there was one LSTM layer and a total of 128 hidden layers, the data was shuffled every epoch for a maximum of 125 epochs, verbose was equal to 0, and the sequence was padded to the left. The GRU had one GRU layer and 150 total hidden layers, the data was shuffled every epoch for a total of 150 epochs, verbose was equal to zero and the sequence was padded to the left. The 5-fold cross-validated results (datapoints were not randomized before creating the folds so as to preserve the timeseries order) appear in Table 2. The GP is most accurate, followed by the LSTM and the GRU. Figure 1 shows the predictions of the GP and the LSTM for the upper tank for one of the test folds. The GP predictions are superior near the sharp inflection and critical points where nonlinearities are strongest. Note that the first 50 points of each test set, which were provided to the LSTM and GRU as a start-up set in the prediction phase, were removed from the calculation of error for both methods.

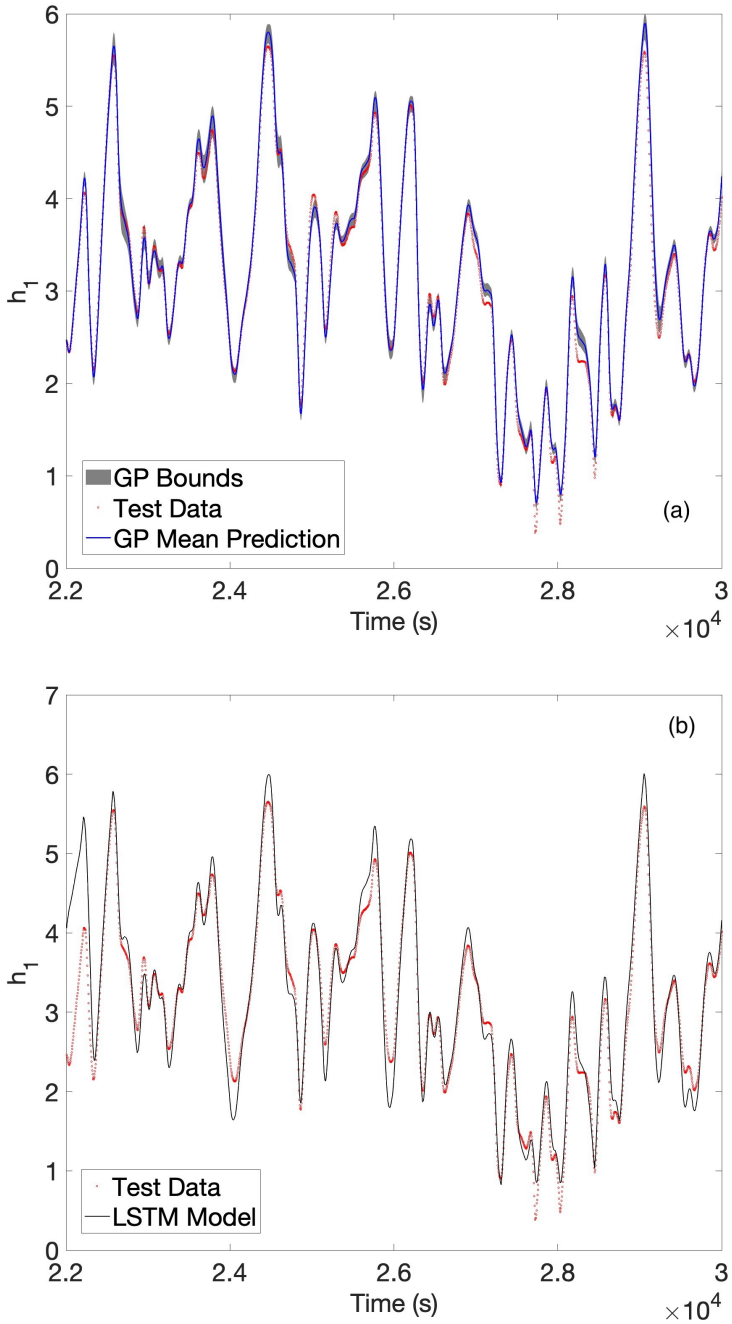

Figure 1: (a) BSS-ANOVA and (b) LSTM predictions vs. test set data for the water level height in tank 1 of the cascaded tanks dataset. Shaded regions in (a) are 95% confidence bounds as estimated from a draw of 40 curves.

3.3 Synthetic dataset: susceptible, infected, recovered model

The susceptible, infected, recovered model (SIR model) is a common simulation for infectious disease. Though there are several versions, the simplest is three states, only two of which are independent. The system is written

$$\dot{S} = -BIS/N_P \tag{18}$$

$$\dot{I} = BIS/N_P - \gamma I \tag{19}$$

$$\dot{R} = \gamma I \tag{20}$$

$$S + I + R = N_P \tag{21}$$

where $S(t)$ is the susceptible population, $I(t)$ the infected, $R(t)$ the recovered, $B(t)$ is the transmissibility rate (which we utilize as a forcing function), γ is the recovery rate (which we leave fixed at 0.5) and N_P is the total population. Because N_P is fixed, only two states are independent, so the system dynamics can be captured by modeling only two of the three. We chose $I(t)$ and $R(t)$.

The training data consists of 58 curves. All curves in the training set have a fixed B value ranging from 0.5 to 9, in six intervals of 1.7. For each value of B there are 8-10 siumulations corresponding to different initial conditions designed in such a way to provide coverage of the state space. (Exact initial conditions used appear in the supplement.) Each simulation used $N_P = 1000$.

The test data consists of 24 curves, each of which features a temporally changing transmissibility $B(t)$. There are three initial B_0 values: 1.35, 4.75 and 8.15. For each starting point there are two types of transmissibility curves: a ramp and a sinusoid. The $B_0 = 1.35$ and $B_0 = 4.75$ starting points have ramps with a positive slope of 1, while the $B_0 = 8.15$ curves have a slope of -1. All ramps run from t=0 to t=4, where they level off. The sinusoids have amplitudes between 0.5 and 3 and a period of 1.

Hyperparameters for the GP were: $a = a_\tau = 4$ for both states, $b_{\tau,R} = 8.95$ and $b_{\tau,I} = 72.1$, while $b_I = 1.25$ and $b_R = 20$. 2000 draws were taken and the first 1000 discarded. The tolerance was 6. Hyperparameters for the LSTM and GRU were the same as for the Cascaded Tanks.

A partial display of the results are shown in Figures 2 for the GP and 3 for the GRU, which was the better performing of the two neural nets on this dataset. For the GP, the total test set MAE was 5.2739±4.0138 for I and 11.8345±21.7337 for R, corresponding to MAPEs of 8.99±4.92 for I and 2.80±2.52 for R. Statistics were not calculated for the GRU as it failed to replicate the dynamics in most test cases and was obviously inferior in a quantitative sense in every instance, as shown in 3.

3.4 Training and inference times

Training and inference times for the GP were fast, with a mean total train time of 6.3 seconds for the cascaded tanks and 10.8 seconds for the SIR, with 8,000 and 20,000 training data points, respectively, on a 2019 6-core i7 processor with 16 GB of RAM. The routines were implemented in MATLAB, but not parallelized or optimized for speed. Models for $\dot{h}_1$ contain between 23 and 41 terms, while $\dot{h}_2$ has between 38 and 57 terms. Prediction times for 2000 static points for the cascaded tanks averages 0.5437 s, and the time for evaluating integrals over the test set averages 20.22 s. For the SIR model the $\dot{I}$ model had 81 terms and the $\dot{R}$ model 9 terms, with a mean integration time of 5.3 s.

Neural networks were native MATLAB functions, parallelized and optimized for speed. Nonetheless average train times were considerably longer, with mean train times of 130 s for the RNN and 175 and 123 s, respectively, for training the LSTM and GRU for the cascaded tanks.

Analyses have shown that the rate limiting step in the GP build algorithms are the $\mathcal{O}(NP)$ construction of the X matrix from the inputs and basis functions.

4 Discussion

The two examples presented in this paper were the only two attempted for dynamic systems identification. Other static prediction problems – especially those involving high dimensionality in the input space with some categorical inputs – are more challenging (the routines perform reasonably well for the 'Adult Income' benchmark for instance but can be exceeded in accuracy by some deep

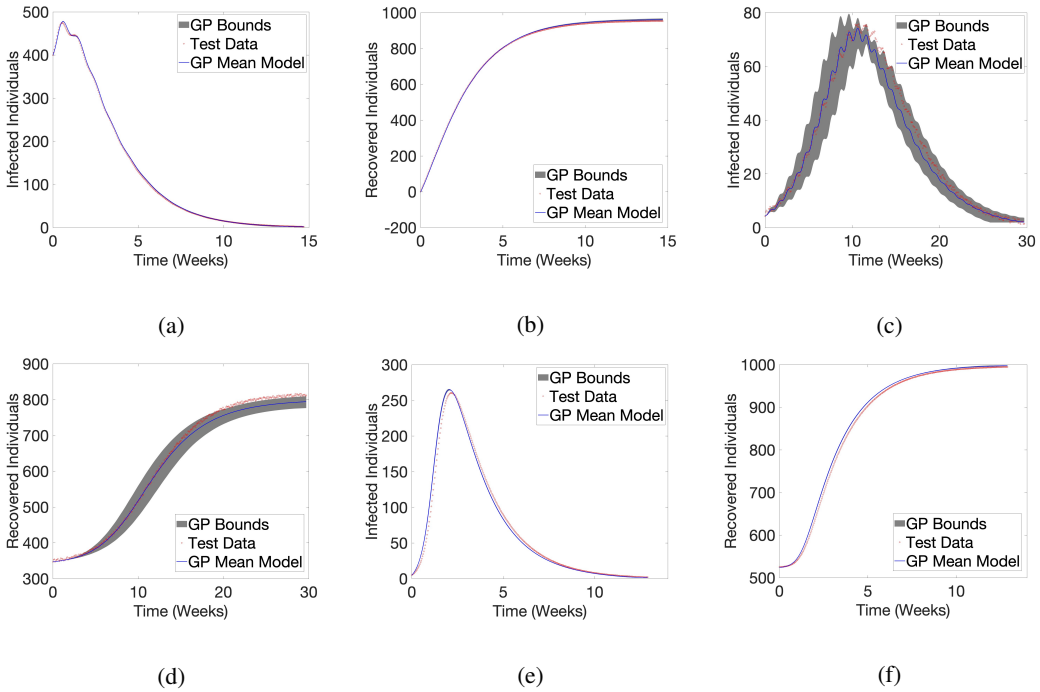

Figure 2: BSS-ANOVA results for 3 curves in the test set: (a)-(b) sine wave transmissibility with low initial infections; (c)-(d) sine wave transmissibility with moderate initial infections; (e)-(f) ramp transmissibility. Shaded regions are 95% confidence bounds for the predictions as estimated from a draw of 40 curves.

learning frameworks and gradient boosted random forests) and require more sophisticated methods for variable selection. Future work will address prediction in these large tabular datasets. Many more experiments and comparisons need to be performed for dynamic systems as well, with larger datasets and more difficult identification problems. The GP is inaccurate in extrapolation: when test set inputs exceed the bounds of the training set the resulting extrapolation sometimes causes instabilities causing the integration procedure to fail. These failures were eliminated by preventing extrapolation even in instances where the inputs exceeded the bounds, but more stable extrapolation strategies will possibly become necessary for longer prediction windows where extrapolation is unavoidable.

Given the performance and speed of the GP for dynamic system identification, control applications are of particular interest. Fast methods for online updating are possible and should make the method a strong contender for these applications.

Optimization and parallelization of theroutines could significantly improve training and inference times. Python-based open-source versions of the optimized codes will be made publicly available.

5 Conclusion

A new forward variable selection algorithm has made the scalable Gaussian process BSS-ANOVA a fast and accurate method for nonparametric regression of tabular data on continuous input spaces. The speed and accuracy for this type of dataset makes it ideal for dynamic system identification. Favorable comparisons with popular neural network approaches for timeseries problems were made on the basis of both speed and accuracy for nonlinear synthetic and benchmark experimental dynamic datasets.

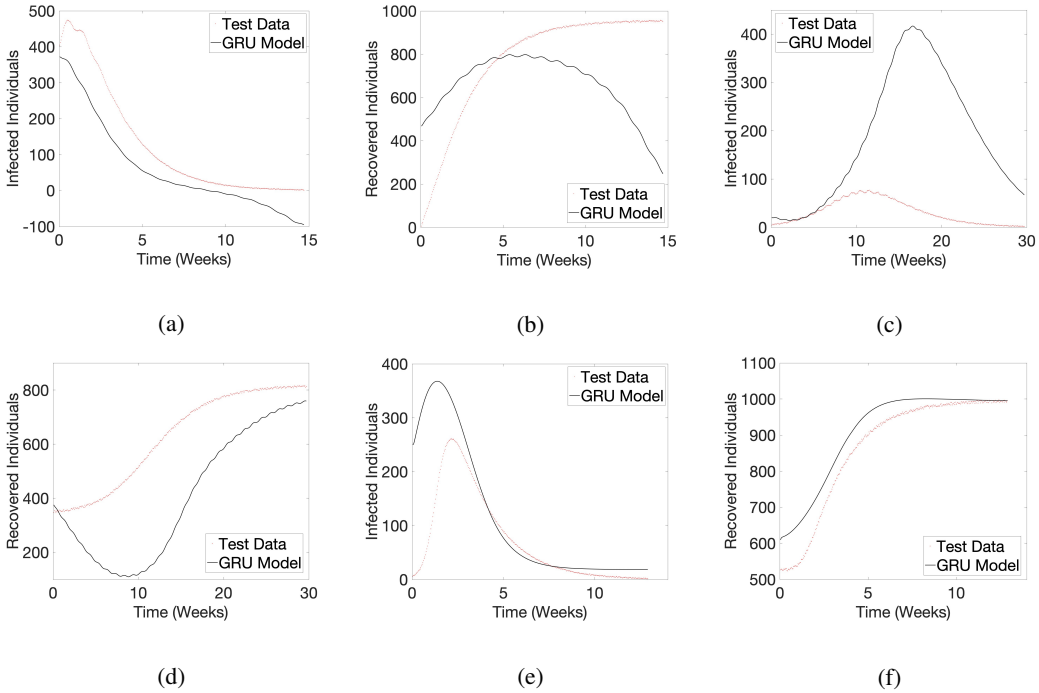

Figure 3: GRU results for 3 curves in the test set: (a)-(b) sine wave transmissibility with low initial infections; (c)-(d) sine wave transmissibility with moderate initial infections; (e)-(f) ramp transmissibility

Acknowledgements

References

K. Sham Bhat, David S. Mebane, Priyadarshi Mahapatra, and Curtis B. Storlie. Upscaling uncertainty with dynamic discrepancy for a multi-scale carbon capture system. *Journal of the American Statistical Association*, 112(520):1453–1467, 2017.

Krzysztof Chalupka, Christopher KI Williams, and Iain Murray. A framework for evaluating approximation methods for Gaussian process regression. *Journal of Machine Learning Research*, 14:333–350, 2013.

Marc Deisenroth and Jun Wei Ng. Distributed Gaussian processes. In *International Conference on Machine Learning*, pages 1481–1490. PMLR, 2015.

MC Kennedy and A O'Hagan. Bayesian calibration of computer models. *Journal of the Royal Statistical Society Series B – Statistical Methodology*, 63(3):425–450, 2001.

Yinkai Lei, Tian-Le Chen, David S. Mebane, and You-Hai Wen. Reduced-order model for microstructure evolution prediction in the electrodes of solid oxide fuel cell with dynamic discrepancy reduced modeling. *Journal of Power Sources*, 416:37–49, 2019.

Haitao Liu, Yew-Soon Ong, Xiaobo Shen, and Jianfei Cai. When gaussian process meets big data: A review of scalable gps. *IEEE Transactions on Neural Networks and Learning Systems*, 31(11): 4405–4423, 2020.

Anca Ostace, Yu-Yen Chen, Robert Parker, David S. Mebane, Chinedu O. Okoli, Andrew Lee, Andrew Tong, Liang-Shih Fan, Lorenz T. Biegler, Anthony P. Burgard, David C. Miller, and Debangsu Bhattacharyya. Kinetic model development and Bayesian uncertainty quantification for the complete reduction of Fe-based oxygen carriers with CH_4, CO and H_2 for chemical looping combustion. *Chemical Engineering Science*, 252117512, 2022.

Joaquin Quinonero-Candela and Carl Edward Rasmussen. A unifying view of sparse approximate Gaussian process regression. *The Journal of Machine Learning Research*, 6:1939–1959, 2005.

Carl Rasmussen and Zoubin Ghahramani. Infinite mixtures of Gaussian process experts. *Advances in Neural Information Processing Systems*, 14, 2001.

Brian J. Reich, Curtis B. Storlie, and Howard D. Bondell. Variable selection in Bayesian smoothing spline ANOVA models: Application to deterministic computer codes. *Technometrics*, 51(2): 110–120, 2009.

M. Schoukens and J. P. Noel. Three benchmarks addressing open challenges in nonlinear system identification. *IFAC Papers Online*, 50(1):446–451, 2017. 20th World Congress of the International-Federation-of-Automatic-Control (IFAC), Toulouse, France, Jul 09-14, 2017.

Liwei Wang, Suraj Yerramilli, Akshay Iyer, Daniel Apley, Ping Zhu, and Wei Chen. Scalable Gaussian processes for data-driven design using big data with categorical factors. *Journal of Mechanical Design*, 144(2), 2022.

Torbjorn Wigren and Johan Schoukens. Three free data sets for development and benchmarking in nonlinear system identification. In *2013 European Control Conference (ECC)*, pages 2933–2938. European Control Assoc; IFAC; IEEE; CSS, 2013. European Control Conference (ECC), ETH Zurich, Zurich, Switzerland, Jul 17-19, 2013.

Andrew Gordon Wilson, Christoph Dann, and Hannes Nickisch. Thoughts on massively scalable Gaussian processes. *arXiv preprint arXiv:1511.01870*, 2015.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] see Section 4
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See the supplement
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Sections 3.2 and 3.3
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Sections 3.2 and 3.3
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section 3.4
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Section 3.2
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] See the supplement
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]