The Gaussian Process Autoregressive Regression Model (GPAR)

James Requeima[12] **Will Tebbutt**[12] **Wessel Bruinsma**[12] **Richard E. Turner**[1]
[1]University of Cambridge and [2]Invenia Labs, Cambridge, UK
{jrr41, wct23, wpb23, ret26}@cam.ac.uk

Abstract

Multi-output regression models must exploit dependencies between outputs to maximise predictive performance. The application of Gaussian processes (GPs) to this setting typically yields models that are computationally demanding and have limited representational power. We present the Gaussian Process Autoregressive Regression (GPAR) model, a scalable multi-output GP model that is able to capture nonlinear, possibly input-varying, dependencies between outputs in a simple and tractable way: the product rule is used to decompose the joint distribution over the outputs into a set of conditionals, each of which is modelled by a standard GP. GPAR's efficacy is demonstrated on a variety of synthetic and real-world problems, outperforming existing GP models and achieving state-of-the-art performance on established benchmarks.

1 Introduction

The Gaussian process (GP) probabilistic modelling framework provides a powerful and popular approach to nonlinear single-output regression (Rasmussen and Williams, 2006). The popularity of GP methods stems from their modularity, tractability, and interpretability: it is simple to construct rich, nonlinear models by compositional covariance function design, which can then be evaluated in a principled way (e.g. via the marginal likelihood), before being interpreted in terms of their component parts. This leads to an attractive plug-and-play approach to modelling and understanding data, which is so robust that it can even be automated (Duvenaud et al., 2013; Sun et al., 2018).

Most regression problems, however, involve multiple outputs rather than a single one. When modelling such data, it is key to capture the dependencies between these outputs. For example, noise in the output space might be correlated, or, whilst one output might depend on the inputs in a complex (deterministic) way, it may depend quite simply on other output variables. In both cases multi-output GP models are required. There is a plethora of existing multi-output GP models that can capture linear correlations between output variables if these correlations are fixed across the input space (Goovaerts, 1997; Wackernagel, 2003; Teh and Seeger, 2005; Bonilla et al., 2008; Nguyen and Bonilla, 2014; Dai et al., 2017). However, one of the main reasons for the popularity of the GP approach is that a suite of different types of nonlinear *input* dependencies can be modelled, and it is disappointing that this flexibility is not extended to interactions between the *outputs*. There are some approaches that do allow limited modelling of nonlinear output dependencies (Wilson et al., 2012; Bruinsma, 2016) but this flexibility comes from sacrificing tractability, with complex and computationally demanding approximate inference and learning schemes now required. This complexity significantly slows down model fitting, evaluation, and improvement work flow.

What is needed is a flexible and analytically tractable modelling approach to multi-output regression that supports plug-and-play modelling and model interpretation. The Gaussian Process Autoregressive Regression (GPAR) model, introduced in Section 2, achieves these aims by taking an approach analogous to that employed by the Neural Autoregressive Density Estimator (Larochelle and Murray, 2011) for density modelling. The product rule is used to decompose the distribution of the outputs given the inputs into a set of one-dimensional conditional distributions. Critically, these distributions can be interpreted as a decoupled set of single-output regression problems, and learning and inference in GPAR therefore amount to a set of standard single-output GP regression tasks: training is closed form, fast, and amenable to standard scaling techniques. GPAR converts the modelling of output dependencies that are possibly nonlinear and input-dependent into a set of standard GP covariance function design problems, constructing expressive, jointly non-Gaussian models over the outputs. Importantly, we show how GPAR can capture nonlinear relationships between outputs as well as structured, input-dependent noise, simply through

Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS) 2019, Naha, Okinawa, Japan. PMLR: Volume 89.

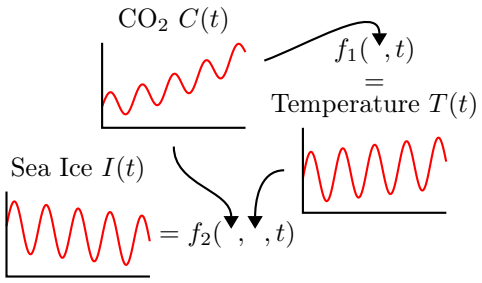

Figure 1: Cartoon motivating a factorisation for the joint distribution $p(I(t), T(t), C(t))$

kernel hyperparameter learning. We apply GPAR to a wide variety of multi-output regression problems, achieving state-of-the-art results on five benchmark tasks.

2 GPAR

Consider the problem of modelling the world's average CO_2 level $C(t)$, temperature $T(t)$, and Arctic sea ice extent $I(t)$ as a function of time t. By the greenhouse effect, one can imagine that the temperature $T(t)$ is a complicated, stochastic function f_1 of CO_2 and time: $T(t) = f_1(C(t), t)$. Similarly, one might hypothesise that the Arctic sea ice extent $I(t)$ can be modelled as another complicated function f_2 of temperature, CO_2, and time: $I(t) = f_2(T(t), C(t), t)$. These functional relationships are depicted in Figure 1 and motivate the following model where the conditionals model the postulated underlying functions f_1 and f_2:

$$\begin{aligned} &p(I(t), T(t), C(t)) \\ &= p(C(t))\, \underbrace{p(T(t) \,|\, C(t))}_{\text{models } f_1}\, \underbrace{p(I(t) \,|\, T(t), C(t))}_{\text{models } f_2}. \end{aligned}$$

More generally, consider the problem of modelling M outputs $y_{1:M}(x) = (y_1(x), \ldots, y_M(x))$ as a function of the input x. Applying the product rule yields[1]

$$\begin{aligned} &p(y_{1:M}(x)) \qquad (1) \\ &= p(y_1(x))\, \underbrace{p(y_2(x) \,|\, y_1(x))}_{\substack{y_2(x) \text{ as a random} \\ \text{function of } y_1(x)}} \cdots \underbrace{p(y_M(x) \,|\, y_{1:M-1}(x))}_{\substack{y_M(x) \text{ as a random} \\ \text{function of } y_{1:M-1}(x)}}, \end{aligned}$$

which states that $y_1(x)$, like CO_2, is first generated from x, according to some unknown, random function f_1; that $y_2(x)$, like temperature, is then generated from $y_1(x)$ and x, according to some unknown, random function f_2; that $y_3(x)$, like the Arctic sea ice extent, is then generated from $y_2(x)$, $y_1(x)$, and x, according to some unknown, random function f_3; *et cetera*:

$$\begin{aligned} y_1(x) &= f_1(x), & f_1 &\sim p(f_1), \\ y_2(x) &= f_2(y_1(x), x), & f_2 &\sim p(f_2), \\ &\vdots \\ y_M(x) &= f_M(y_{1:M-1}(x), x) & f_M &\sim p(f_M). \end{aligned}$$

GPAR, introduced now, models these unknown functions $f_{1:M}$ with Gaussian processes (GPs). Recall that a GP f over an index set $\mathcal{X}$ defines a stochastic process, or process in short, where $f(x_1), \ldots, f(x_N)$ are jointly Gaussian distributed for any $x_1, \ldots, x_N$ (Rasmussen and Williams, 2006). Marginalising out $f_{1:M}$, we find that GPAR models the conditionals in Equation (1) with Gaussian processes:

$$\begin{aligned} &y_m \,|\, y_{1:m-1} \qquad (2) \\ &\quad \sim \mathcal{GP}(0, k_m((y_{1:m-1}(x), x), (y_{1:m-1}(x'), x'))). \end{aligned}$$

Although the conditionals in Equation (1) are Gaussian, the joint distribution $p(y_{1:N})$ is not; moments of the joint distribution over the outputs are generally intractable, but samples can be generated by sequentially sampling the conditionals. Figure 2b depicts the graphical model corresponding to GPAR. Crucially, the kernels (k_m) may specify nonlinear, input-dependent relationships between outputs, which enables GPAR to model data where outputs inter-depend in complicated ways.

Returning to the climate modelling example, one might object that the temperature $T(t)$ at time t does not just depend the CO_2 level $C(t)$ at time t, but instead depends on the entire history of CO_2 levels C: $T(t) = f_1(C, t)$. *Note: by writing C instead of $C(t)$, we refer to the entire function C, as opposed to just its value at t.* Similarly, one might object that the Arctic sea ice extent $I(t)$ at time t does not just depend on the temperature $T(t)$ and CO_2 level $C(t)$ at time t, but instead depends on the entire history of temperatures T and CO_2 levels C: $T(t) = f_2(T, C, t)$. This kind of dependency structure, where output $y_m(x)$ now depends on the entirety of all foregoing outputs $y_{1:m}$ instead of just their value at x, motivates the following generalisation of GPAR in its form of Equation (2):

$$y_m \,|\, y_{1:m-1} \sim \mathcal{GP}(0, k_m((y_{1:m-1}, x), (y_{1:m-1}, x'))), \quad (3)$$

which we refer to as *nonlocal* GPAR, or *non-instantaneous* in the temporal setting. Clearly, GPAR is a special case of nonlocal GPAR. Figure 2a depicts the graphical model corresponding to nonlocal GPAR. Nonlocal GPAR will not be evaluated experimentally, but some theoretical properties will be described.

Inference and learning in GPAR. Inference and learning in GPAR is simple. Let $y_m^{(n)} = y_m(x^{(n)})$ denote an observation for output m. Then, assuming all

[1] This requires an ordering of the outputs; we will address this point in Section 2.

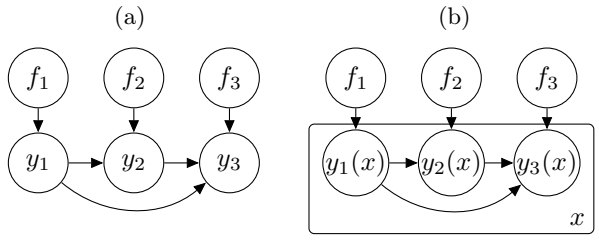

Figure 2: Graphical models corresponding to (a) non-local GPAR and (b) GPAR

outputs are observed at each input, we find

$$p(f_{1:M} \mid (y^{(n)}_{1:M}, x^{(n)})_{n=1}^N) \tag{4}$$
$$= \prod_{m=1}^{M} p(f_m \mid \underbrace{(y^{(n)}_m)_{n=1}^N}_{\substack{\text{observations} \\ \text{for } f_m}}, \underbrace{(y^{(n)}_{1:m-1}, x^{(n)})_{n=1}^N}_{\substack{\text{input locations} \\ \text{of observations}}}),$$

meaning that the posterior of f_m is computed simply by conditioning f_m on the observations for output m located at the observations for the foregoing outputs and the input, which again is a Gaussian process; thus, like the prior, the posterior predictive density also decomposes as a product of Gaussian conditionals. The evidence $\log p((y^{(n)}_{1:M})_{n=1}^N \mid (x^{(n)})_{n=1}^N)$ decomposes similarly; as a consequence, the hyperparameters for f_m can be learned simply by maximising $\log p((y^{(n)}_m)_{n=1}^N \mid (y^{(n)}_{1:m-1}, x^{(n)})_{n=1}^N)$. In conclusion, *inference and learning in GPAR with M outputs comes down to inference and learning in M decoupled, one-dimensional GP regression problems.* This shows that without approximations GPAR scales $\mathcal{O}(MN^3)$, depending only linearly on the number of outputs M rather than cubically, as is the case for general multi-output GPs, which scale $\mathcal{O}(M^3N^3)$.

Scaling GPAR. In these one-dimensional GP regression problems, off-the-shelf GP scaling techniques may be applied to trivially scale GPAR to large data sets. Later we utilise the variational inducing point method by Titsias (2009), which is theoretically principled (Matthews et al., 2016) and empirically accurate (Bui et al., 2016b). This method requires a set of inducing inputs to be specified for each Gaussian process. For the first output y_1 there is a single input x, which is time. We use fixed (non-optimised) regularly spaced inducing inputs, as they are known to perform well for time series (Bui and Turner, 2014). The second and following outputs y_m, however, require inducing points to be placed in x *and* $y_1, \ldots, y_{m-1}$. Regular spacing can be used again for x, but there are choices available for $y_1, \ldots, y_{m-1}$. One approach would be to optimise these inducing input locations, but instead we use the posterior predictive means of $y_1, \ldots, y_{m-1}$ at x. This choice accelerates training and was found to yield good results.

Missing data. When there are missing outputs, the procedures for inference and learning described above remain valid as long as for every observation $y^{(n)}_m$ there are also observations $y^{(n)}_{m-1:1}$. We call a data set satisfying this property *closed downwards*. If a data set is not closed downwards, data may be imputed, e.g. using the model's posterior predictive mean, to ensure closed downwardness; inference and learning then, however, become approximate. The key conditional independence expressed by Figure 2b that results in simple inference and learning is the following:

Theorem 1. Let a set of observations $\mathcal{D}$ be closed downwards. Then $y_i \perp \mathcal{D}_{i+1:M} \,|\, \mathcal{D}_{1:i}$, where $\mathcal{D}_{1:i}$ are the observations for outputs $1, \ldots, i$ and $\mathcal{D}_{i+1:M}$ for $i+1, \ldots, M$.

Theorem 1 is proved in Appendix A from the supplementary material. Note that Theorem 1 holds for every graphical model of the form of Figure 2b, meaning that the decomposition into single-output modelling problems holds for *any* choice of the conditionals in Equation (1), not just Gaussians.

Potential deficiencies of GPAR. GPAR has two apparent limitations. First, since the outputs from earlier dimensions are used as inputs to later dimensions, noisy outputs yield noisy inputs. One possible mitigating solution is to employ a *denoising* input transformation for the kernels, e.g. using the posterior predictive mean of the foregoing outputs as the input of the next covariance function. We shall refer to GPAR models employing this approach as D-GPAR. Second, one needs to choose an ordering of the outputs. Fortunately, often the data admits a natural ordering; for example, if the predictions concern a single output dimension, this should be placed last. Alternatively, if there is not a natural ordering, one can greedily optimise the evidence with respect to the ordering. This procedure considers $\frac{1}{2}M(M+1)$ configurations while an exhaustive search would consider all $M!$ possible configurations: the best first output is selected out of all M choices, which is then fixed; then, the best second output is selected out of the remaining $M-1$ choices, which is then fixed; et cetera. These methods are examined in Section 6.

3 GPAR and Multi-Output GPs

The choice of kernels $k_{1:M}$ for $f_{1:M}$ is crucial to GPAR, as they determine the types of relationships between inputs and outputs that can be learned. Particular choices for $k_{1:M}$ turn out to yield models closely related to existing work. These connections are made rigorous by the nonlinear and linear equivalent model discussed in Appendix B from the supplementary material. We

summarise the results here; see also Table 1.

If k_m depends linearly on the foregoing outputs $y_{1:m-1}$ at particular x, then a joint Gaussian distribution over the outputs is induced in the form of a multi-output GP model (Goovaerts, 1997; Stein, 1999; Wackernagel, 2003; Teh and Seeger, 2005; Bonilla et al., 2008; Nguyen and Bonilla, 2014) where latent processes are mixed together according to a matrix, called the *mixing matrix*, that is *lower-triangular* (Appendix B). One may let the dependency of k_m on $y_{1:m-1}$ vary with x, in which case the mixing matrix varies with x, meaning that correlations between outputs vary with x. This yields an instance of the Gaussian Process Regression Network (GPRN) (Wilson et al., 2012) where inference is fast and closed form. One may even let k_m depend nonlinearly on $y_{1:m-1}$, which yields a particularly structured deep Gaussian process (DGP) (Damianou, 2014; Bui et al., 2016a), potentially with skip connections from the inputs (Appendix B). Note that GPAR may be interpreted as a conventional DGP where the hidden layers are directly observed and correspond to successive outputs; this connection could potentially be leveraged to bring machinery developed for DGPs to GPAR, e.g. to deal with arbitrarily missing data.

One can further let k_m depend on the *entirety* of the foregoing outputs $y_{1:m-1}$, yielding instances of nonlocal GPAR. An example of a nonlocal linear kernel is

$$k((y,x),(y',x')) = \int a(x-z,x'-z')y(z)y'(z')\,\mathrm{d}z\,\mathrm{d}z'.$$

The nonlocal linear kernel again induces a jointly Gaussian distribution over the outputs in the form of a convolutional multi-output GP model (Álvarez et al., 2009; Álvarez and Lawrence, 2009; Bruinsma, 2016) where latent processes are convolved together according to a matrix-valued function, called the *convolution matrix*, that is *lower-triangular* (Appendix B). Again, one may let the dependency of k_m on the entirety of $y_{1:m-1}$ vary with x, in which case the convolution matrix varies with x, or even let k_m depend nonlinearly on the entirety of $y_{1:m-1}$; an example of a nonlocal nonlinear kernel is

$$k(y,y') = \sigma^2 \exp\left(-\int \frac{1}{2\ell(z)}(y(z)-y'(z))^2\,\mathrm{d}z\right).$$

Henceforth, we shall refer to GPAR with linear dependencies between outputs as GPAR-L, GPAR with nonlinear dependencies between outputs as GPAR-NL, and a combination of the two as GPAR-L-NL.

4 Further Related Work

The Gaussian Process Network (Friedman and Nachman, 2000) is similar to GPAR, but was instead developed to identify causal dependencies between variables in a probabilistic graphical models context rather than multi-output regression. The work by Yuan (2011) also discusses a model similar to GPAR, but specifies a different generative procedure for the outputs.

The multi-fidelity modelling literature is closely related to multi-output modelling. Whereas in a multi-output regression task we predict all outputs, multi-fidelity modelling is concerned with predicting a particular high-fidelity function, incorporating information from observations from various levels of fidelity. The idea of iteratively conditioning on lower fidelity models in the construction of higher fidelity ones is a well-used strategy (Kennedy and O'Hagan, 2000; Le Gratiet and Garnier, 2014). The model presented by Perdikaris et al. (2017) is nearly identical to GPAR applied in the multi-fidelity framework, but applications outside this setting have not been considered.

Moreover, GPAR follows a long line of work on the family of fully visible Bayesian networks (Frey et al., 1996; Bengio and Bengio, 2000) that decompose the distribution over the observations according to the product rule (Equation (1)) and model the resulting one dimensional conditionals. A number of approaches use neural networks for this purpose (Neal, 1992; Frey et al., 1996; Larochelle and Murray, 2011; Theis and Bethge, 2015; van den Oord et al., 2016). In particular, if the observations are real-valued, a standard architecture lets the conditionals be Gaussian with means encoded by neural networks and fixed variances. Under broad conditions, if these neural networks are replaced by Bayesian neural networks with independent Gaussian priors over the weights, we recover GPAR as the width of the hidden layers goes to infinity (Neal, 1996; Matthews et al., 2018).

5 Synthetic Data Experiments

GPAR is well suited for problems where there is a strong functional relationship between outputs and for problems where observation noise is richly structured and input dependent. In this section we demonstrate GPAR's ability to model both types of phenomena.

First, we test the ability to leverage strong functional relationships between the outputs. Consider three outputs y_1, y_2, and y_3, inter-depending nonlinearly:

$$\begin{aligned}
y_1(x) &= -\sin(10\pi(x+1))/(2x+1) - x^4 + \epsilon_1,\\
y_2(x) &= \cos^2(y_1(x)) + \sin(3x) + \epsilon_2,\\
y_3(x) &= y_2(x)y_1^2(x) + 3x + \epsilon_3,
\end{aligned}$$

where $\epsilon_1, \epsilon_2, \epsilon_3 \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, 0.05)$. By substituting y_1 and y_2 into y_3, we see that y_3 can be expressed directly in terms of x, but via a complex function. The dependence of y_3 on y_1, y_2, however, is much simpler. Therefore, as GPAR can exploit direct dependencies between $y_{1:2}$ and

GPAR	Deps. Between Outputs	Kernels for $f_{1:M}$	Related Models
Local	Linear	$k_1(x,x') + k_{\text{Linear}}(y(x), y(x'))$	Multi-Output GPs [1]
	+ dep. on x	$k_1(x,x') + k_2(x,x')k_{\text{Linear}}(y(x), y(x'))$	GPRN [2]
	Nonlinear	$k_1(x,x') + k_2(y(x), y(x'))$	Deep GPs (DGPs) [3]
	+ dep. on x	$k_1(x,x') + k_2((x,y(x)),(x',y(x')))$	DGPs with input connections
Nonlocal	Linear	$k_1(x,x') + k_{\text{Linear}}(y,y')$	
	+ dep. on x	$k_1(x,x') + k_2(x,x')k_{\text{Linear}}(y,y')$	Convolutional MOGPs [4]
	Nonlinear	$k_1(x,x') + k_2(y,y')$	
	+ dep. on x	$k_1(x,x') + k_2((x,y),(x',y'))$	

Table 1: Classification of kernels $k_{1:M}$ for $f_{1:M}$, the resulting dependencies between outputs, and related models. Here k_{Linear} refers to a linear kernel and k_1 and k_2 to an exponentiated quadratic (EQ) or rational quadratic (RQ) kernel (Rasmussen and Williams, 2006). [1]: Goovaerts (1997); Stein (1999); Wackernagel (2003); Teh and Seeger (2005); Bonilla et al. (2008); Osborne et al. (2008); Nguyen and Bonilla (2014). [2]: Wilson et al. (2012). [3]: Damianou (2014); Bui et al. (2016a). [4]: Álvarez et al. (2009); Álvarez and Lawrence (2009); Bruinsma (2016).

y_3, it should be presented with a much simplified task as compared to predicting y_3 from x directly. Figure 3 shows plots of independent GPs (IGPs) and GPAR fit to 30 data points from y_1, y_2 and y_3. Indeed observe that GPAR is able to learn y_2's dependence on y_1, and y_3's dependence on y_1 and y_2, whereas the independent GPs struggle with the complicated structure.

Second, we test GPAR's ability to capture non-Gaussian and input-dependent noise. Consider the following three schemes in which two outputs are observed under various noise conditions: $y_1(x) = f_1(x) + \epsilon_1$ and

$$\begin{aligned}
&(1):\ y_2(x) = f_2(x) + \sin^2(2\pi x)\epsilon_1 + \cos^2(2\pi x)\epsilon_2,\\
&(2):\ y_2(x) = f_2(x) + \sin(\pi\epsilon_1) + \epsilon_2,\\
&(3):\ y_2(x) = f_2(x) + \sin(\pi x)\epsilon_1 + \epsilon_2,
\end{aligned}$$

where $\epsilon_1, \epsilon_2 \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, 0.1)$, and f_1 and f_2 are complicated, nonlinear functions.[2] All three schemes have i.i.d. homoscedastic Gaussian noise in y_1. The noise in y_2, however, depends on that in y_1 and can be heteroscadastic. The task for GPAR is to learn the scheme's noise structure. Figure 4 visualises the noise correlations induced by the schemes and the noise structures learned by GPAR. Observe that GPAR is able to learn the various noise structures.

6 Real-World Data Experiments

In this section we evaluate GPAR's performance and compare to other models on four standard data sets commonly used to evaluate multi-output models. We also consider a recently-introduced data set in the field of Bayesian optimisation, which is a downstream application area that could benefit from GPAR. Table 2 lists the models against which we compare GPAR. We always compare against IGP and CK, ICM, SLFM, and CGP, and compare against CMOGP and GPRN if results for the considered task are available. Since CK and ICM are much simplified versions of SLFM (Álvarez et al., 2010; Goovaerts, 1997) and CGP is an approximation to SLFM, we sometimes omit results for CK, ICM, and CGP. Implementations can be found at `https://github.com/wesselb/gpar` (Python) and `https://github.com/willtebbutt/GPAR.jl` (Julia). Experimental details can be found in Appendix D from the supplementary material.

Acronym	Model
IGP	Independent GPs
CK	Cokriging
ICM	Intrinstic Coregionalisation Model [1]
SLFM	Semi-Parametric Latent Factor Model [2]
CGP	Collaborative Multi-Output GPs [3]
CMOGP	Convolved Multi-Output GP Model [4]
GPRN	GP Regression Network [5]

Table 2: Models against which GPAR is compared. [1]: Goovaerts (1997); Stein (1999); Wackernagel (2003). [2]: Teh and Seeger (2005). [3]: Nguyen and Bonilla (2014). [4]: Álvarez and Lawrence (2011); Álvarez et al. (2010). [5]: Wilson et al. (2012).

Electroencephalogram (EEG) data set.[3] This data set consists of 256 voltage measurements from 7 electrodes placed on a subject's scalp whilst the subject is shown a certain image; Zhang et al. (1995) describe the data collection process in detail. In particular, we use frontal electrodes FZ and F1–F6 from the first trial on control subject 337. The task is to predict the last 100 samples for electrodes FZ, F1, and F2, given that the first 156 samples of FZ, F1, and F2

[2] The functions given by $f_1(x) = -\sin(10\pi(x+1))/(2x+1) - x^4$ and $f_2(x) = \frac{1}{5}e^{2x}(\theta_1\cos(\theta_2\pi x) + \theta_3\cos(\theta_4\pi x)) + \sqrt{2x}$.

[3] The EEG data set can be downloaded at `https://archive.ics.uci.edu/ml/datasets/eeg+database`.

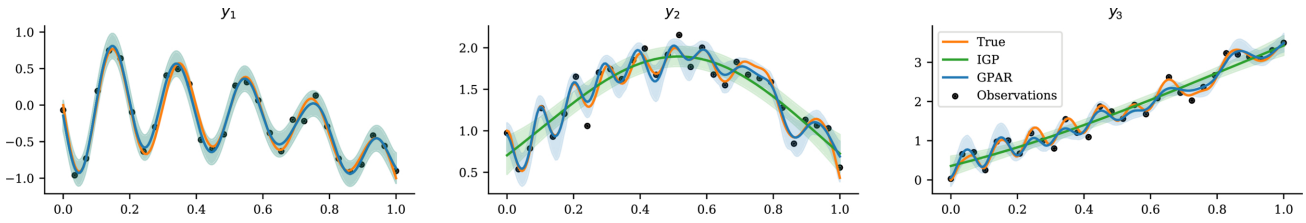

Figure 3: Synthetic data set with complex output dependencies: GPAR vs independent GPs (IGP) predictions.

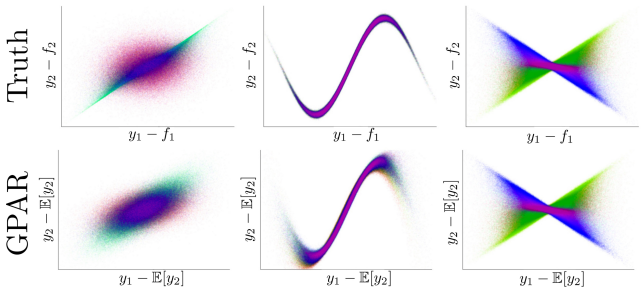

Figure 4: Correlation between the sample residues (deviation from the mean) for y_1 and y_2. Left, middle, and right plots correspond to schemes (1), (2) and (3) respectively. Samples are coloured according to input value x; that is, all samples for a particular x have the same colour. If the colour pattern is preserved, then GPAR has successfully captured how the noise in y_1 correlates to that in y_2.

Model	SMSE	MLL	TT
IGP	1.75	2.60	2 sec
SLFM	1.06	4.00	11 min
GPAR-NL	**0.26**	**1.63**	5 sec

Table 3: Results for the EEG data set for IGP, the SLFM with four latent dimensions, and GPAR.

and the whole signals of F3–F6 are observed. Performance is measured with the standardised mean squared error (SMSE), mean log loss (MLL) (Rasmussen and Williams, 2006), and training time (TT). Figure 5 visualises predictions for electrode F2, and Table 3 quantifies the results. We observe that GPAR-NL outperforms independent in terms of SMSE and MLL; note that independent GPs completely fail to provide an informative prediction. Furthermore, independent GPs were trained in two seconds, and GPAR-NL took only three more seconds; in comparison, training SLFM took 11 minutes.

Jura data set.[4] This data set comprises metal concentration measurements collected from the topsoil in a 14.5 km^2 region of the Swiss Jura. We follow the experimental protocol by Goovaerts (1997) also followed by Álvarez and Lawrence (2011): The training data comprises 259 data points distributed spatially with three output variables—nickel, zinc, and cadmium—and 100 additional data points for which only two of the three outputs—nickel and zinc—are observed. The task is to predict cadmium at the locations of those 100 additional data. Performance is evaluated with the mean absolute error (MAE).

[4]The data can be downloaded at `https://sites.google.com/site/goovaertspierre/pierregoovaertswebsite/download/`.

Model	IGP	CK†	ICM	SLFM	CMOGP†
MAE	0.5739	0.51	0.4601	0.4606	0.4552
MAE*	0.5753		0.4114	0.4145	

Model	GPRN†	GPAR-NL	D-GPAR-NL
MAE	0.4525	0.4324	**0.4114**
MAE*	0.4040	0.4168	**0.3996**

Table 4: Results for the Jura data set for IGP, cokriging (CK) and ICM with two latent dimensions, the SLFM with two latent dimensions, CMOGP, GPRN, and GPAR. * Results are obtained by first log-transforming the data, then performing prediction, and finally transforming the predictions back to the original domain. † Results from Wilson (2014).

Table 4 shows the results. The comparatively poor performance of independent GPs highlights the importance of exploiting correlations between the mineral concentrations. Furthermore, Table 4 shows that D-GPAR-NL significantly outperforms the other models, achieving a new state-of-the-art.

Exchange rates data set.[5] This data set consists of the daily exchange rate w.r.t. USD of the top ten international currencies (CAD, EUR, JPY, GBP, CHF, AUD, HKD, NZD, KRW, and MXN) and three precious metals (gold, silver, and platinum) in the year 2007. The task is to predict CAD on days 50–100, JPY on days 100–150, and AUD on days 150–200, given that CAD is observed on days 1–49 and 101–251, JPY on days 1–99 and 151–251, and AUD on days 1–149 and 201–251; and that all other currencies are observed throughout the whole year. Performance is measured

[5]The exchange rates data set can be downloaded at `http://fx.sauder.ubc.ca`.

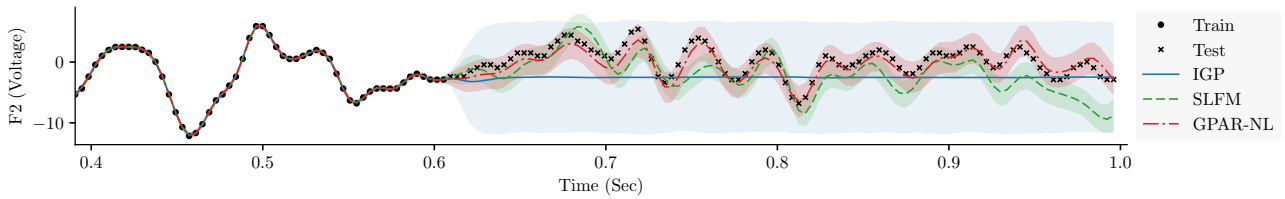

Figure 5: Predictions for electrode F2 from the EEG data set

Model	IGP*	CMOGP*	CGP*	GPAR-L-NL
SMSE	0.5996	0.2427	0.2125	**0.0302**

Table 5: Experimental results for the exchange rates data set for IGP, CMOGP, CGP, and GPAR. * These numbers are taken from Nguyen and Bonilla (2014).

with the SMSE.

Figure 6 visualises GPAR's prediction for data set, and Table 5 quantifies the result. We greedily optimise the evidence w.r.t. the ordering of the outputs to determine an ordering, and we impute missing data to ensure closed downwardness of the data. Observe that GPAR significantly outperforms all other models.

Tidal height, wind speed, and air temperature data set.[6] This data set was collected at 5 minute intervals by four weather stations: Bramblemet, Cambermet, Chimet, and Sotonmet, all located in Southampton, UK. The task is to predict the air temperature measured by Cambermet and Chimet from all other signals. Performance is measured with the SMSE. This experiment serves two purposes. First, it demonstrates that it is simple to scale GPAR to large data sets using off-the-shelf inducing point techniques for single-output GP regression. Second, it shows that scaling to large data sets enables GPAR to better learn dependencies between outputs, which, importantly, can significantly improve predictions in regions where outputs are partially observed. We utilise the variational inducing point method by Titsias (2009) as discussed in Section 2, with 10 inducing points per day. This data set is not closed downwards, so we use mean imputation when training. We use D-GPAR-L and set the temporal kernel to be a simple EQ, meaning that the model *cannot* make long-range predictions, but instead must exploit correlations between outputs.

Nguyen and Bonilla (2014) consider from this data set 5 days in July 2013, and predict short periods of the air temperature measured by Cambermet and Chimet using all other signals. We followed their setup and predicted the same test set, but instead trained on the *whole* of July. Even though the additional observations do not temporally correlate with the test periods at all, they enable GPAR to better learn the relationships between the outputs, which, unsurprisingly, significantly improved the predictions: using the whole of July, GPAR achieves SMSE 0.056, compared to their SMSE 0.107.

The test set used by Nguyen and Bonilla (2014) is rather small, yielding high-variance test results. We therefore do not pursue further comparisons on their train–test split, but instead consider a bigger, more challenging setup: using as training data 10 days (days $[10, 20)$, roughly 30 k points), 15 days (days $[18, 23)$, roughly 47 k points), and the whole of July (roughly 98 k points), make predictions of 4 day periods of the air temperature measured by Cambermet and Chimet. Figure 7 visualises the test periods and GPAR's predictions for it. Despite the additional observations not correlating with the test periods, we observe clear, though dimishining, improvements in the predictions as the training data is increased.

MLP validation error data set. The final data set is the validation error of a multi-layer perceptron (MLP) on the MNIST data, trained using categorical cross-entropy, and set as a function of six hyperparameters: the number of hidden layers, the number of neurons per hidden layer, the dropout rate, the learning rate to use with the ADAM optimizer, the L_1 weight penalty, and the L_2 weight penalty. This experiment was implemented using code made available by Hernández-Lobato (2016). An improved model for the objective surface could translate directly into improved performance in Bayesian optimisation (Snoek et al., 2012), as this would enable a more informed search of the hyperparameter space.

To generate a data set, we sample 291 sets of hyperparameters randomly from a rectilinear grid and train the MLP for 21 epochs under each set of hyperparameters, recording the validation performance after 1, 5, 11, 16, and 21 epochs. We construct a training set of 175 of these hyperparameter settings and, crucially, discard roughly 30% of the validation performance results at 5 epochs at random, and again discard roughly 30% of those results at 11 epochs, and so forth. The resulting data set has 175 labels after 1 epoch, 124 after 5, 88

[6] The data can be downloaded at `http://www.bramblemet.co.uk`, `http://cambermet.co.uk`, `http://www.chimet.co.uk`, and `http://sotonmet.co.uk`.

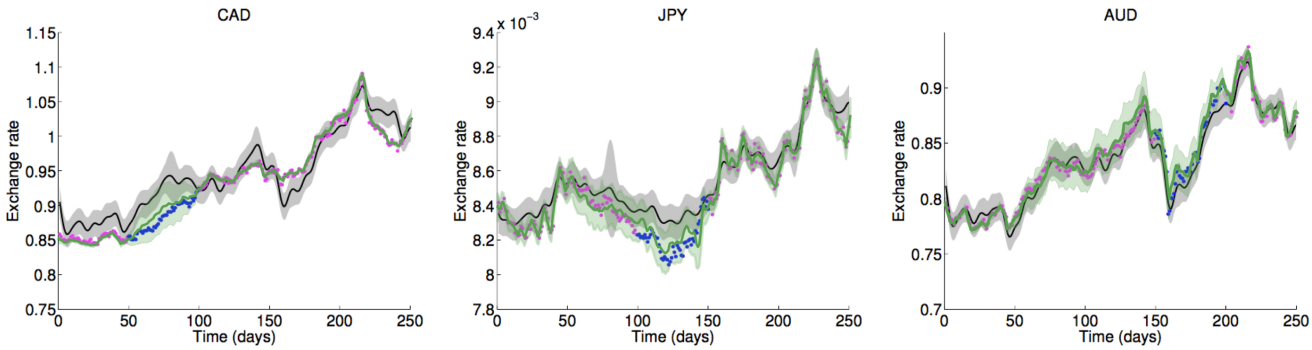

Figure 6: Visualisation of the exchange rates data set and CGP's (black) and GPAR's (green) predictions for it. GPAR's predictions are overlayed on the original figure by Nguyen and Bonilla (2014).

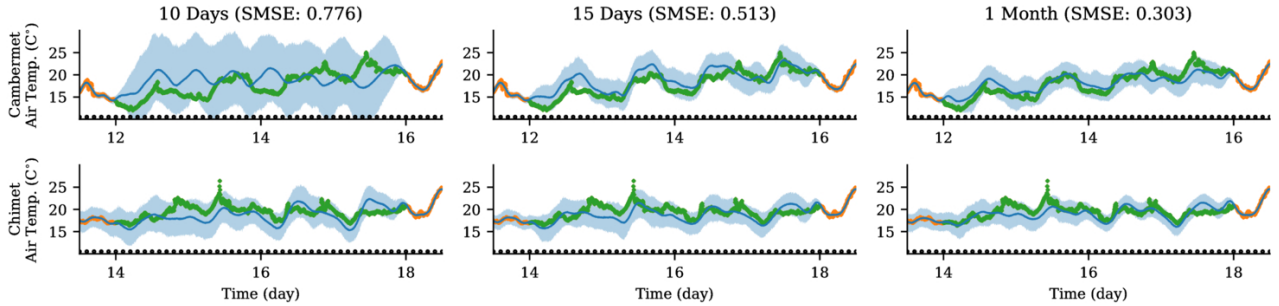

Figure 7: Visualisation of the air temperature data set and GPAR's prediction for it. Black circles indicate the locations of the inducing points.

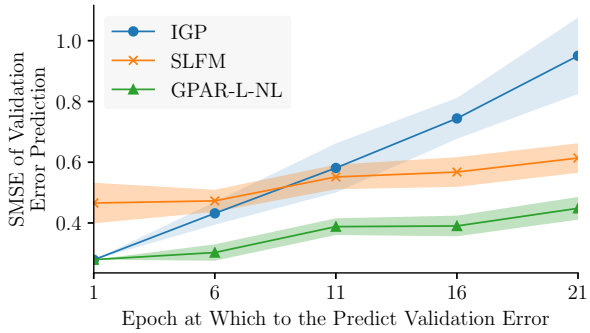

Figure 8: Results for the machine learning data set for a GP, the SLFM with two latent dimensions, and GPAR

after 11, 64 after 15 and 44 after 21, simulating the partial completion of the majority of runs. Importantly, a Bayesian Optimisation system typically exploits only completed training runs to inform the objective surface, whereas GPAR can also exploit partially complete runs.

The results presented in Figure 8 show the SMSE in predicting validation performance at each epoch using GPAR, the SLFM, and independent GPs on the test set, averaged over 10 seed for the pseudo-random number generator used to select which outputs from the training set to discard. GPs trained independently to predict performance after a particular number of epochs perform worse than the SLFM and GPAR, which both have learned to exploit the extra information available. In turn, GPAR noticeably outperforms the SLFM.

7 Conclusion and Future Work

This paper introduced GPAR: a flexible, fast, tractable, and interpretable approach to multi-output GP regression. GPAR can model (1) nonlinear relationships between outputs and (2) complex output noise. GPAR can scale to large data sets by trivially leveraging scaling techniques for one-dimensional GP regression (Titsias, 2009). In effect, GPAR transforms high-dimensional data modelling problems into set of single-output modelling problems, which are the bread and butter of the GP approach. GPAR was rigorously tested on a variety of synthetic and real-world problems, consistently outperforming existing GP models for multi-output regression. An exciting future application of GPAR is to use compositional kernel search (Lloyd et al., 2014) to automatically learn and explain dependencies between outputs and inputs. Further insights into structure of the data could be gained by decomposing GPAR's posterior over additive kernel components (Duvenaud, 2014). These two approaches could be developed into a useful tool for automatic structure discovery. Two further exciting future applications of GPAR are modelling of environmental phenomena and improving data efficiency of existing Bayesian optimisation tools (Snoek et al., 2012).

Acknowledgements

Richard E. Turner is supported by Google as well as EPSRC grants EP/M0269571 and EP/L000776/1.

References

Álvarez, M. and Lawrence, N. D. (2009). Sparse convolved gaussian processes for multi-output regression. *Advances in Neural Information Processing Systems*, 21:57–64.

Álvarez, M. A. and Lawrence, N. D. (2011). Computationally efficient convolved multiple output Gaussian processes. *Journal of Machine Learning Research*, 12:1459–1500.

Álvarez, M. A., Luengo, D., and Lawrence, N. D. (2009). Latent force models. *Artificial Intelligence and Statistics*, 5:9–16.

Álvarez, M. A., Luengo, D., Titsias, M. K., and Lawrence, N. D. (2010). Efficient multioutput Gaussian processes through variational inducing kernels. *Journal of Machine Learning Research: Workshop and Conference Proceedings*, 9:25–32.

Bengio, Y. and Bengio, S. (2000). Modeling high-dimensional discrete data with multi-layer neural networks. In *Advances in Neural Information Processing Systems*, pages 400–406.

Bonilla, E. V., Chai, K. M., and Williams, C. K. I. (2008). Multi-task Gaussian process prediction. *Advances in Neural Information Processing Systems*, 20:153–160.

Bruinsma, W. P. (2016). The generalised gaussian convolution process model. MPhil thesis, Department of Engineering, University of Cambridge.

Bui, T. D., Hernández-Lobato, D., Li, Y., Hernández-Lobato, J. M., and Turner, R. E. (2016a). Deep gaussian processes for regression using approximate expectation propagation. *arXiv preprint arXiv:1602.04133*.

Bui, T. D. and Turner, R. E. (2014). Tree-structured Gaussian process approximations. *Advances in Neural Information Processing Systems*, 27:2213–2221.

Bui, T. D., Yan, J., and Turner, R. E. (2016b). A unifying framework for gaussian process pseudo-point approximations using power expectation propagation. *arXiv preprint arXiv:1605.07066*.

Dai, Z., Álvarez, M. A., and Lawrence, N. D. (2017). Efficient modeling of latent information in supervised learning using Gaussian processes. *arXiv preprint arXiv:1705.09862*.

Damianou, A. (2014). *Deep Gaussian Processes and Variational Propagation of Uncertainty*. PhD thesis, Department of Neuroscience, University of Sheffield.

Duvenaud, D. (2014). *Automatic Model Construction with Gaussian Processes*. PhD thesis, Computational and Biological Learning Laboratory, University of Cambridge.

Duvenaud, D., Lloyd, J. R., Grosse, R., Tenenbaum, J. B., and Ghahramani, Z. (2013). Structure discovery in nonparametric regression through compositional kernel search. In *International Conference on Machine Learning*.

Frey, B. J., Hinton, G. E., and Dayan, P. (1996). Does the wake-sleep algorithm produce good density estimators? In *Advances in neural information processing systems*, pages 661–667.

Friedman, N. and Nachman, I. (2000). Gaussian process networks. In *Uncertainty in Artificial Intelligence*, pages 211–219. Morgan Kaufmann Publishers Inc.

Goovaerts, P. (1997). *Geostatistics for Natural Resources Evaluation*. Oxford University Press, 1 edition.

Hernández-Lobato, J. M. (2016). Neural networks with optimal accuracy and speed in their predictions.

Kennedy, M. C. and O'Hagan, A. (2000). Predicting the output from a complex computer code when fast approximations are available. *Biometrika*, 87(1):1–13.

Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.

Larochelle, H. and Murray, I. (2011). The neural autoregressive distribution estimator. In *AISTATS*, volume 1, page 2.

Le Gratiet, L. and Garnier, J. (2014). Recursive co-kriging model for design of computer experiments with multiple levels of fidelity. *International Journal for Uncertainty Quantification*, 4(5).

Lloyd, J. R., Duvenaud, D., Grosse, R., Tenenbaum, J. B., and Ghahramani, Z. (2014). Automatic construction and natural-language description of nonparametric regression models. In *Association for the Advancement of Artificial Intelligence (AAAI)*.

Matthews, A. G. D. G., Hensman, J., Turner, R. E., and Ghahramani, Z. (2016). On sparse variational methods and the kullback-leibler divergence between stochastic processes. *Artificial Intelligence and Statistics*, 19.

Matthews, A. G. d. G., Rowland, M., Hron, J., Turner, R. E., and Ghahramani, Z. (2018). Gaussian process behaviour in wide deep neural networks. *arXiv preprint arXiv:1804.11271*.

Neal, R. M. (1992). Connectionist learning of belief networks. *Artificial intelligence*, 56(1):71–113.

Neal, R. M. (1996). *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media.

Nguyen, T. V. and Bonilla, E. V. (2014). Collaborative multi-output Gaussian processes. *Conference on Uncertainty in Artificial Intelligence*, 30.

Nocedal, J. and Wright, S. J. (2006). *Numerical Optimization*. Springer, second edition.

Osborne, M. A., Roberts, S. J., Rogers, A., Ramchurn, S. D., and Jennings, N. R. (2008). Towards real-time information processing of sensor network data using computationally efficient multi-output Gaussian processes. In *Proceedings of the 7th International Conference on Information Processing in Sensor Networks*, IPSN '08, pages 109–120. IEEE Computer Society.

Perdikaris, P., Raissi, M., Damianou, A., Lawrence, N. D., and Karniadakis, G. E. (2017). Nonlinear information fusion algorithms for data-efficient multi-fidelity modelling. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science*, 473.

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. MIT Press.

Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, pages 2951–2959.

Stein, M. (1999). *Interpolation of Spatial Data*. Springer-Verlag New York, 1 edition.

Sun, S., Zhang, G., Wang, C., Zeng, W., Li, J., and Grosse, R. (2018). Differentiable compositional kernel learning for gaussian processes. *International Conference on Machine Learning*, 35.

Teh, Y. W. and Seeger, M. (2005). Semiparametric latent factor models. *International Workshop on Artificial Intelligence and Statistics*, 10.

Theis, L. and Bethge, M. (2015). Generative image modeling using spatial lstms. In *Advances in Neural Information Processing Systems*, pages 1927–1935.

Titsias, M. K. (2009). Variational learning of inducing variables in sparse Gaussian processes. *Artificial Intelligence and Statistics*, 12:567–574.

van den Oord, A., Kalchbrenner, N., Espeholt, L., Vinyals, O., Graves, A., et al. (2016). Conditional image generation with pixelcnn decoders. In *Advances in Neural Information Processing Systems*, pages 4790–4798.

Wackernagel, H. (2003). *Multivariate Geostatistics*. Springer-Verlag Berlin Heidelberg, 3 edition.

Wilson, A. G. (2014). *Covariance Kernels for Fast Automatic Pattern Discovery and Extrapolation With Gaussian Processes*. PhD thesis, University of Cambridge.

Wilson, A. G., Knowles, D. A., and Ghahramani, Z. (2012). Gaussian process regression networks. *International Conference on Machine Learning*, 29.

Yuan, C. (2011). Conditional multi-output regression. In *International Joint Conference on Neural Networks*, pages 189–196. IEEE.

Zhang, X., Begleiter, H., Porjesz, B., Wang, W., and Litke, A. (1995). Event related potentials during object recognition tasks. *Brain Research Bulletin*, 38(6):531–538.

A Conditional Independence in Figure 2b

In this section, we prove the key conditional independence in Figure 2b that makes GPAR work:

Theorem 2. Let a set of observations $\mathcal{D}$ be closed downwards. Then $y_i \perp \mathcal{D}_{i+1:M} \,|\, \mathcal{D}_{1:i}$, where $\mathcal{D}_{1:i}$ are the observations for outputs $1, \ldots, i$ and $\mathcal{D}_{i+1:M}$ for $i+1, \ldots, M$.[7]

To begin with, we review some basic notions concerning graphical models. Let a path be a sequence of nodes $v_1, \ldots, v_n$ from some directed graph G where, for each v_i, G either contains an edge from v_i to v_{i+1}, or an edge from v_{i+1} to v_i. If G contains an edge from a to b, then write $a \to b$ to mean the two-node path in which node b follows node a. Similarly, if G contains an edge from b to a, then write $a \leftarrow b$ to mean the two-node path in which b follows a. Write $a \rightleftharpoons b$ to mean either $a \to b$ or $b \leftarrow a$.

Definition 1 (Active Path (Definition 3.6 from Koller and Friedman (2009)))**.** Let $\mathcal{P} = v_1 \rightleftharpoons \cdots \rightleftharpoons v_n$ be a path in a graphical model. Let Z be a subset of the variables from the graphical model. Then, call $\mathcal{P}$ active given Z if (1) for every v-structure $v_{i-1} \to v_i \leftarrow v_{i+1}$ in $\mathcal{P}$, v_i or a descendant of v_i is in Z; and (2) no other node in $\mathcal{P}$ is in Z.

Definition 2 (d-Separation (Definition 3.7 from Koller and Friedman (2009)))**.** Let X, Y, and Z be three sets of nodes from a graphical model. Then, call X and Y d-separated given Z if no path between any $x \in X$ and $y \in Y$ is active given Z.

Theorem 3 (d-Separation Implies Conditional Independence (Theorem 3.3 from Koller and Friedman (2009)))**.** Let X, Y, and Z be three sets of nodes from a graphical model. If X and Y are d-separated given Z, then $X \perp Y \,|\, Z$.

Define the *layer* of a node in Figure 2b to be

$$\text{layer}(f_i) = \text{layer}(y_i(x)) = i.$$

We are now ready to prove Theorem 2.

Proof of Theorem 2. For $i < j$, let $\mathcal{P}$ be a path between any $y_i(x') \in y_i$ and $y_j(x) \in \mathcal{D}_{i+1:N}$. Let $y_k(\hat{x})$ be the first node in $\mathcal{P}$ such that $\text{layer}(y_k(\hat{x})) > i$. Then $\mathcal{P}$ contains

$$\cdots \to y_m(\hat{x}) \to y_k(\hat{x}) \rightleftharpoons \cdots$$

for some $m \leq i < k$.

If $y_k(\hat{x}) \in \mathcal{D}_{i+1:N}$, then, since $\mathcal{D}$ is closed downwards, $y_m(\hat{x}) \in \mathcal{D}_{1:i}$, meaning that $\mathcal{P}$ is inactive.

If, on the other hand, $y_k(\hat{x}) \notin \mathcal{D}$, then, since $\mathcal{D}$ is closed downwards, $y_{k'}(\hat{x}) \notin \mathcal{D}$ for all $k' \geq k$. Therefore, $y_j(x)$ cannot be descendant of $y_k(\hat{x})$, so $\mathcal{P}$ must contain

$$\cdots \to y_{k'}(\hat{x}) \to y_{k''}(\hat{x}) \leftarrow f_{k''} \to \cdots$$

for some $m \leq k' < k''$, which forms a v-structure. We conclude that $\mathcal{P}$ is inactive, because $y_{k''}(\hat{x})$ nor a descendant of $y_{k''}(\hat{x})$ can be in $\mathcal{D}$. □

[7] $\mathcal{D}_{1:i} = \{y_j^{(n)} \in \mathcal{D} : j \leq i, n \leq N\}$ and $\mathcal{D}_{i+1:M} = \{y_j^{(n)} \in \mathcal{D} : j > i, n \leq N\}$.

B The Nonlinear and Linear Equivalent Model

In this section, we construct equivalent models for GPAR (Lemmas 1 and 2). These models make GPAR's connection to other models in the literature explicit.

To begin with, we must introduce some notation and definitions. For functions $A, B \colon \mathcal{X} \times (\mathcal{Y}^M)^{\mathcal{X}} \to \mathcal{Y}^M$, define composition $\circ$ as follows: $(A \circ B)(x, y) = A(x, B(\cdot, y))$. Note that $\circ$ is well-defined and associative. For a function $u \colon \mathcal{X} \to \mathcal{Y}^M$, denote $A \circ u \colon \mathcal{X} \to \mathcal{Y}^M$, $A \circ u = A(\cdot, u)$. Again, note that $(A \circ B) \circ u = A \circ (B \circ u)$. Furthermore, denote

$$\underbrace{A \circ \cdots \circ A}_{n \text{ times}} = A^n.$$

Consider a function $A \colon \mathcal{X} \times (\mathcal{Y}^M)^{\mathcal{X}} \to \mathcal{Y}^M$ such that $A_i(x, y) \colon \mathcal{X} \times (\mathcal{Y}^M)^{\mathcal{X}} \to \mathcal{Y}$ depends only on $(x, y_{1:i-1})$, where $A_1 = 0$. Further let $u, y \colon \mathcal{X} \to \mathcal{Y}^M$, denote $\mathrm{T}\, f = u + A \circ f$, and denote N consecutive applications of T by T^N.

The expression $\mathrm{T}^{M-1}\, u$ will be key in constructing the equivalent models. We show that it is the unique solution of a functional equation:

Proposition 1. The unique solution of $y = u + A \circ y$ is $y = \mathrm{T}^{M-1}\, u$.

Proof of Proposition 1. First, we show that $y = u + A \circ y$ has a solution, and that this solution is unique. Because $A_i(x, y)$ depends only on $(x, y_{1:i-1})$, it holds that

$$y_i = u_i + A_i \circ y = u_i + A_i \circ (y_{1:i-1}, 0),$$

where $(y_{1:i-1}, 0)$ represents the concatenation of $y_{1:i-1}$ and $M - i + 1$ zeros. Thus, y_i can uniquely be constructed from u_i, A_i, and $y_{1:i-1}$; therefore, y_1 exists and is unique, so y_2 exists and is unique: by induction we find that y exists and is unique.

Second, we show that $y = \mathrm{T}^{M-1} u$ satisfies $y = u + A \circ y = \mathrm{T} y$. To show this, we show that $(\mathrm{T}^n u)_i = (\mathrm{T}^{n-1} u)_i$ for $i = 1, \ldots, n$, for all n. To begin with, we show the base case, $n = 1$:

$$(\mathrm{T} u)_1 = u_1 + A_1 \circ u = u_1 = (\mathrm{T}^0 u)_1,$$

since $A_1 = 0$. Finally, suppose that the claim holds for a particular n. We show that the claim then holds for $n+1$: Let $i \leq n+1$. Then

$$\begin{aligned}
(\mathrm{T}^{n+1} u)_i &= u_i + A_i \circ \mathrm{T}^n u \\
&= u_i + A_i \circ ((\mathrm{T}^n u)_{1:i-1}, (\mathrm{T}^n u)_{i:M}) \\
&= u_i + A_i \circ ((\mathrm{T}^{n-1} u)_{1:i-1}, (\mathrm{T}^n u)_{i:M}) \\
&\qquad\qquad \text{(By assumption)} \\
&\overset{(*)}{=} u_i + A_i \circ ((\mathrm{T}^{n-1} u)_{1:i-1}, (\mathrm{T}^{n-1} u)_{i:M}) \\
&= u_i + A_i \circ \mathrm{T}^{n-1} u \\
&= (\mathrm{T}^n u)_i,
\end{aligned}$$

where $(*)$ holds because $A_i(x, y)$ depends only on $(x, y_{1:i-1})$. □

In the linear case, $\mathrm{T}^{M-1} u$ turns out to greatly simplify.

Proposition 2. If $A(x, y)$ is linear in y, then $\mathrm{T}^{M-1} u = (\sum_{i=0}^{M-1} A^i) \circ u$.

Proof of Proposition 2. If $A(x, y)$ is linear in y, then one verifies that $\circ$ distributes over addition. Therefore,

$$\begin{aligned}
\mathrm{T}^{M-1} u &= u + A \circ \mathrm{T}^{M-2} u \\
&= u + A \circ u + A^2 \circ \mathrm{T}^{M-3} u \\
&\vdots \\
&= u + A \circ u + \cdots + A^{M-1} \circ u. \qquad \square
\end{aligned}$$

We now use Propositions 1 and 2 to construct a nonlinear and linear equivalent model.

Lemma 1 (Nonlinear Equivalent Model)**.** Let A be an M-dimensional vector-valued process over $\mathcal{X} \times (\mathcal{Y}^M)^{\mathcal{X}}$, each A_i drawn from $\mathcal{GP}(0, k_{A_i})$ independently, and let u be an M-dimensional vector-valued process over $\mathcal{X}$, each u_i drawn from $\mathcal{GP}(0, k_{u_i})$ independently. Furthermore, let $A_i(x, y) \colon \mathcal{X} \times (\mathcal{Y}^M)^{\mathcal{X}} \to \mathcal{Y}$ depend only on $(x, y_{1:i-1})$, meaning that $k_{A_i} = k_{A_i}(x, y_{1:i-1}, x', y'_{1:i-1})$, and let $A_1 = 0$. Denote $\mathrm{T} f = u + A \circ f$, and denote N consecutive applications of T by T^N. Then

$$\begin{gathered}
y \,|\, A, u = \mathrm{T}^{M-1} u \iff \\
y_i \,|\, y_{1:i-1} \sim \mathcal{GP}(0, k_{u_i} + k_{A_i}(\,\cdot\,, y_{1:i-1}, \,\cdot\,, y_{1:i-1})).
\end{gathered}$$

Proof of Lemma 1. Since $A_i(x, y)$ depends only on $(x, y_{1:i-1})$, it holds by Proposition 1 that any sample from $y \,|\, A, u$ satisfies $y_i = u_i + A_i \circ y$, so $y_i = u_i + A_i \circ (y_{1:i-1}, 0)$, where $(y_{1:i-1}, 0)$ represents the concatenation of $y_{1:i-1}$ and $M - i + 1$ zeros. The equivalence now follows. □

Lemma 2 (Linear Equivalent Model)**.** Suppose that A was instead generated from

$$A(x, y) \,|\, \hat{A} = \int \hat{A}(x - z) y(z) \,\mathrm{d}z,$$

where $\hat{A}$ is an $(M \times M)$-matrix-valued process over $\mathcal{X}$, each $\hat{A}_{i,j}$ drawn from $\mathcal{GP}(0, k_{\hat{A}_{i,j}})$ independently if $i > j$ and $\hat{A}_{i,j} = 0$ otherwise. Then

$$\begin{gathered}
y \,|\, A, u = \left(\sum_{i=0}^{M-1} A^i \right) \circ u \iff \\
y_i \,|\, y_{1:i-1} \sim \mathcal{GP}(0, k_{u_i} + k_{A_i}(\,\cdot\,, y_{1:i-1}, \,\cdot\,, y_{1:i-1})), \quad (5)
\end{gathered}$$

where

$$\begin{aligned}
&k_{A_i}(x, y_{1:i-1}, x', y'_{1:i-1}) \\
&\quad = \sum_{j=1}^{i-1} \int k_{\hat{A}_{i,j}}(x - z, x' - z') y_j(z) y'_j(z') \,\mathrm{d}z \,\mathrm{d}z'.
\end{aligned}$$

Proof of Lemma 2. First, one verifies that $A_i(x, y)$ still depends only on $(x, y_{1:i-1})$, and that $A_i(x, y)$ is linear in y. The result then follows from Lemma 1 and Proposition 2, where the expression for k_{A_i} follows from straightfoward calculation. □

As mentioned in the paper, the kernels for $f_{1:M}$ determine the types of relationships between inputs and outputs that can be learned. Lemmas 1 and 2 make this explicit: Lemma 1 shows that nonlocal GPAR can recover a model where M latent GPs u are repeatedly composed with another latent GP A, where A has a particular dependency structure, and Lemma 2 shows that nonlocal GPAR can recover a model where M latent GPs u are linearly transformed, where the linear transform $T = \sum_{i=0}^{M-1} A^i$ is lower triangular and may vary with the input.

In Lemma 2, note that it is not restrictive that T is lower triangular: Suppose that T were dense. Then, letting $y \,|\, T, u = T \circ u$, $y \,|\, T$ is jointly Gaussian. Hence $y_i \,|\, y_{1:i-1}, T$ is a GP whose mean linearly depends upon $y_{1:i-1}$ via T, meaning that $y_i \,|\, y_{1:i-1}$ is of the form of Equation (5) where k_{u_i} may be more complicated. If, however, $\hat{A}(z) = \delta(z) B$ for some random $(M \times M)$-matrix B, each $B_{i,j}$ drawn from $\mathcal{N}(0, \sigma^2_{B_{i,j}})$ if $i > j$ and $B_{i,j} = 0$ otherwise, then it is restrictive that T is lower triangular: In this case, $y(x) \,|\, B, u = \sum_{i=0}^{M-1} B^i u(x)$. If $T = \sum_{i=0}^{M-1} B^i$ were dense, then, letting $y \,|\, T, u = Tu$, y can be represented with Lemma 2 if and only if $y \,|\, T$'s covariance can be diagonalised by a constant, invertible,

lower-triangular matrix. This condition does not hold in general, as Lemma 3 proves.

C Lemma 3

Call functions $k_1, \ldots, k_M \colon \mathcal{X} \to \mathbb{R}$ linearly independent if

$$\left(\forall\, x : \sum_{i=1}^{M} c_i k_i(x) = 0 \right) \implies c_1 = \ldots = c_M = 0.$$

Lemma 3. Let $k_1, \ldots, k_M \colon \mathcal{X} \to \mathbb{R}$ be linearly independent and arrange them in a diagonal matrix $K = \mathrm{diag}(k_1, \ldots, k_n)$. Let A be an invertible $M \times M$ matrix such that its columns cannot be permuted into a triangular matrix. Then there does not exist an invertible triangular matrix T such that $T^{-1} B K(x) B^{\mathsf{T}} T^{-\mathsf{T}}$ is diagonal for all x.

Proof. Suppose, on the contrary, that such T does exist. Then two different rows a_p and a_q of $A = T^{-1}B$ share nonzero elements in some columns C; otherwise, A would have exactly one nonzero entry in every column—A is invertible—so A would be the product of a permutation matrix and a diagonal matrix, meaning that $B = TA$'s columns could be permuted into a triangular matrix. Now, by $T^{-1} B K(x) B^{\mathsf{T}} T^{-\mathsf{T}} = A K(x) A^{\mathsf{T}}$ being diagonal for all $x \in \mathcal{X}$, $\sum_i a_{p,i} a_{q,i} k_i(x) = 0$ for all x. Therefore, by linear independence of $k_1, \ldots, k_N$, it holds that $a_{p,i} a_{q,i} = 0$ for all i. But $a_{p,i} a_{q,i} \neq 0$ for any $i \in C$, which is a contradiction. □

D Experimental Details

For every experiment, the form of the kernels is determined by the particular GPAR model used: GPAR-L, GPAR-NL, or GPAR-L-NL (see Table 2 in the main paper), potentially with a D-$*$ prefix to indicate the denoising procedure outlined in "Potential deficiencies of GPAR" in Section 2 of the paper main. For GPAR-NL, we always used exponentiated quadratic (EQ) kernels, except for the exchange rates experiment, where we used rational quadratic (RQ) kernels (Rasmussen and Williams, 2006). Furthermore, in every problem we simply expanded according to Equation (1) in the main paper or greedily optimised the ordering, in both cases putting the to-be-predicted outputs last. We used `scipy`'s implementation of the L-BFGS-B algorithm (Nocedal and Wright, 2006) to optimise hyperparameters.