Multi-layer Stacked Gaussian Processes

Bradley J. Gram-Hansen Stephen J. Roberts
Department of Engineering
University of Oxford

{bradley,sjrob} @ robots.ox.ac.uk

Abstract

In this work we explore the construction of a simple, yet effective, Gaussian process framework for supervised learning. The models that we propose take inspiration from the construction of deep Gaussian processes, as they take the form of a hierarchical multi-layered Gaussian process, with a stacked kernel. We show that with our models it is possible to get enhanced predictions, when compared to a vanilla Gaussian process. We also show, that with our framework it is possible to extract underlying structure from the data, even when using small amounts of data. We then demonstrate our models on a wide variety of data sets, to analyse both the strength and weaknesses of our architectures.

1 Introduction

Gaussian processes (GPs) are flexible and adaptable non-parametric Bayesian models, that are used in a plateau of domains; ranging from finance[1], to natural language processing [2], to modelling astronomical phenomenons [3], to health care [4]. One particular strength of GPs is in their analytically simplicity, which gives us clear insight into how they work and operate. For each point that is predicted with a GPs, we are given the perceived uncertainty of that prediction, which is absolutely essential for any safety critical process, such as those performed in health care [4] or in the context of autonomous robots. [5]

Inspired by the works of Duvenaud et al.[6], Neal [7] and Daminou and Lawrence [8], we propose two simple architectures, which allow us to gain more insight into the structure of our data. The first architecture can be seen as a multi-layered GPs regression approach with a stacked kernel, which we call the single-input model, as at each layer we only pass through the original observations. The second approach, inspired by Duvenaud et al. [6] is multi-layered GPs regression with a stacked kernel, where at each layer we pass through the posterior mean from the previous layer and augment it with the original observations. We call this model the augmented-input model.

In the succeeding sections, we shall give a brief overview of Gaussian Processes section 2, details for the implementation and construction of both the single-input model and the augmented-input model section 3, and present a discussion of results section 4.

2 Gaussian processes and Kernels

In this section we discuss a formal, yet brief, introduction to GPs and in particular Gaussian process regression. In addition to that, we also provide details about different types of kernels, covariance functions, with a particular focus on those that we have used to produce the results presented in this paper. For a more complete introduction to both GPs and kernels see [9].

2.1 Gaussian processes

Despite the name Gaussian, GPs are not confined to modelling data which we believe has been drawn from some underlying Gaussian distribution. The very fact that they are *non-parametric* models allows us to characterise rich types of behaviours due to the potentially infinite parameter space gifted to us by the kernels, subsection 2.2. This, coupled with a Bayesian framework, allows us to perform a central task in supervised machine learning, that is to learn the *function f*, which maps a set of observables $\mathbf{x} \in \mathbb{R}^{n \times D}$ to a set of targets $\mathbf{y} \in \mathbb{R}^{n \times 1}$, where n represents the number of observables and D the dimension of the input data . The functions $f(\mathbf{x})$ are priors, random variables, which are drawn from the following GP:

$$f(\mathbf{x}) \sim GP(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \tag{1}$$

Formally we state that a GP is a collection of random variables, such that any *finite* collection of these f must follow a multivariate Gaussian distribution, which is completely specified by its mean function $\mu(\mathbf{x})$ and its covariance function, the kernel, $k(\mathbf{x}, \mathbf{x})$:

$$\mu(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})] \tag{2}$$

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - \mu(\mathbf{x}))(f(\mathbf{x}') - \mu(\mathbf{x}'))^T] \tag{3}$$

As we are modelling realistic data, we can make a strong assumption that the function values that we observe are noisy and so $y = f(\mathbf{x}) + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma_y^2)$ and σ_y^2 is a hyperparameter representing the noise variance. In order to perform GP regression we would like to select only the functions f, that agree with our observations. Thus, in the noisy scenario, given this set of functions, our observables $\mathbf{x}$, and test inputs $x^* \in \mathbb{R}^{n^* \times D}$, we would like to sample the functions f^* generated from the joint posterior distribution. This corresponds to sampling the posterior mean and covariance from the conditional of the joint prior distribution of the original observed data, augmented with the test inputs and outputs. The joint prior distribution is defined as:

$$P\left(\begin{bmatrix} \mathbf{y} \\ f^* \end{bmatrix}\right) \sim \mathcal{N}\left(\begin{bmatrix} \mu(\mathbf{x}) \\ \mu(x^*) \end{bmatrix}, \begin{bmatrix} k(\mathbf{x}, \mathbf{x}) + \sigma_y^2 \mathbb{I} & k(\mathbf{x}, x^*) \\ k(x^*, \mathbf{x}) & k(x^*, x^*) \end{bmatrix}\right) \tag{4}$$

Thus, conditioning equation 4, which means evaluating $P(f^*|\mathbf{x}, \mathbf{y}, x^*) = \mathcal{N}(\bar{f^*}, \text{covar}(\bar{f^*}))$, allows us to extract the posterior mean, our predictions, and posterior covariance, our uncertainty in those predictions:

$$\bar{f^*} = \mu(x^*) + k(x^*, \mathbf{x})[k(\mathbf{x}, \mathbf{x}) + \sigma_n^2 \mathbb{I}] \tag{5}$$

$$\text{covar}(\bar{f^*}) = k(x^*, x^*) - k(x^*, \mathbf{x})[k(\mathbf{x}, \mathbf{x}) + \sigma_y^2 \mathbb{I}]^{-1} k(\mathbf{x}, x^*) \tag{6}$$

and throughout our models we set the mean function $\mu(\mathbf{x}) = \mathbf{0}$. This means that $\mathbf{y}$ is distributed as $\mathbf{y} \sim \mathcal{N}(\mathbf{0}, k(\mathbf{x}, \mathbf{x}) + \sigma_y^2 \mathbb{I})$ and so the marginal likelihood $P(\mathbf{y}|\mathbf{x})$, which we will require for optimising over the hyperparameters, is given by:

$$P(\mathbf{y}|\mathbf{x}) = -\frac{1}{2}\mathbf{y}^T[k(\mathbf{x}, \mathbf{x}) + \sigma_n^2 \mathbb{I}]^{-1}\mathbf{y} - \frac{1}{2}\log|k(\mathbf{x}, \mathbf{x}) + \sigma_n^2 \mathbb{I}| - \frac{n}{2}\log 2\pi \tag{7}$$

2.2 Kernels

Our choice of kernel and the associated hyperparameters is fundamental to how good a prediction we can make. Having the right kernel, but the wrong hyperparameters or vice versa, will lead to very poor predictions. Throughout this work we use a combination of four different kernels, which we shall describe below. The intialised values for the hyperparameters are selected at random from a uniform distribution on the interval $[a, b]$[1], unless we have strong prior knowledge about our data. The first of the kernels is the squared exponential kernel:

$$k_{SE}(x, x') = h^2 \exp\left(-\frac{(x - x')}{2l^2}\right) \tag{8}$$

where $l \in \mathbb{R}^+$ represents the length scale associated with input x and $h \in \mathbb{R}^+$ defines the vertical scale of variations. The second is the standard periodic kernel [10]:

$$k_{PER}(x, x') = h \exp\left[-\frac{1}{2}\left(\frac{\sin^2(\frac{\pi}{\lambda}|x - x'|)}{l^2}\right)\right] \tag{9}$$

[1] Our interval is typically [0,1].

where λ represents the period. The third is the Matern class of kernels:

$$k_{MAT}(x, x') = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu(x-x')}}{l} \right)^{\nu} J_{\nu} \left(\frac{\sqrt{2\nu(x-x')}}{l} \right) \tag{10}$$

where J_{ν} is a modified Bessel function and $\nu = p + \frac{1}{2}$, where $p \in \mathbb{Z}^+$. The fourth is the neural network kernel (multi-layer perceptron kernel) [11]:

$$k_{MLP}(x, x') = h^2 \frac{2}{\pi} \arcsin \left(\frac{h_w^2 x^T x' + h_b^2}{\sqrt{h_w^2 x^T x + h_b^2 + 1}\sqrt{h_w^2 x'^T x' h_b^2 + 1}} \right) \tag{11}$$

where h_w^2 represents the variances of the prior over the input weights [2] and h_b^2 represents the variance of the prior over the bias parameters.

3 Stacked kernel Gaussian processes

Initially, for comparative purposes we designed a single-input model of the stacked kernel approach. However, to our surprise, it also delivered results which improved upon the vanilla GPs discussed in section 2. Both models are implemented in Python 3 using the *GPy* [12] package.

3.1 Single-input model

The model begins by selecting a kernel and its associated hyperparameters at random, unless we have strong prior knowledge about the data. The model then takes the observed data $\mathbf{x}$ and performs Gaussian process regression to map to the target values $\mathbf{y}$. It then optimises over the hyperparameters of the log marginal likelihood and predicts on the test values. Then, at the next layer l, we select a new kernel at random, add this to the kernel chosen in layer $l-1$ and repeat the steps above as shown in algorithm 1. Therefore, by the time we are in layer i, our kernel will be a linear combination of all previous kernels $K_i = \sum_{l=0}^{l=i} k_l$. Our priors f will now be drawn from $f_i = \sum_{l=0}^{l=i} f_l \sim GP_i(\mathbf{0}, K_i)$. The linear superposition of kernels, *additive kernels* [13], allows us to discover non-local structures and model richer types of behaviour. This is due in part to a larger parameter space, which can be seen by looking at the derivative of the log marginal likelihood equation (13). As our GP is conditioned on the previous kernels the prediction becomes:

$$\bar{f}_i^* = \mathbb{E}[f_i^* | \mathbf{x}, \mathbf{y}_i, x^*, K_i] = K_i(x^*, \mathbf{x})[K_i(\mathbf{x}, \mathbf{x}) + \sigma_y^2 \mathbb{I}]^{-1} \mathbf{y}_i \tag{12}$$

which has exactly the same form of the standard GP, with the exception K. Likewise, the posterior covariance will have the same form as that of the standard GP, except with a different substitution for the kernel and $\mathbf{y}_i$ is distributed as $\mathbf{y}_i \sim \mathcal{N}(\mathbf{0}, K_i(\mathbf{x}, \mathbf{x}) + \sigma_y^2 \mathbb{I})$

$$\frac{\partial}{\partial \theta_j} \log P(\mathbf{y}_i | \mathbf{x}, \theta_i) = \frac{1}{2} \mathbf{y}_i^T K_i^{-1}(\mathbf{x}, \mathbf{x}) \frac{\partial K_i}{\partial \theta_{ij}} K_i^{-1}(\mathbf{x}, \mathbf{x}) \mathbf{y}_i - \frac{1}{2} \text{tr}(K_i^{-1}) \frac{\partial K_i(\mathbf{x}, \mathbf{x})}{\partial \theta_{ij}} \tag{13}$$

Algorithm 1 Single-input stacked GPs regression

Input: $\mathbf{x}, x^*, \mathbf{y}_0, k_0, \theta, N, \sigma_y^2, i$
for $l = 0 : i$ **do**
  if $l > 0$ **then**
    $K_l \leftarrow K_{l-1} + k_{new}$ ▷ k_{new} is a new kernel selected at random
  else
    $K_l \leftarrow k_0$
  $L \leftarrow \text{cholesky}(K_l(\mathbf{x}, \mathbf{x}) + \sigma_y^2)$
  $\alpha \leftarrow L^T(L \backslash \mathbf{y}_l)$
  $\bar{f}_l{}^* \leftarrow K_l(x^*, \mathbf{x})\alpha$ ▷ Posterior mean
  $\gamma \leftarrow L \backslash K_l(\mathbf{x}, x^*)$
  $\text{covar}(\bar{f}_l{}^*) \leftarrow K_l(x^*, x^*) - \gamma^T \gamma$ ▷ Posterior variance
  $\log P(\mathbf{y}_l | \mathbf{x}) \leftarrow -\frac{1}{2} \mathbf{y}_i^T \alpha - Tr(\log L) - \frac{n}{2} \log 2\pi$ ▷ Gaussian likelihood
  $\theta_l \leftarrow optimise \ \log P(\mathbf{y}_l | \mathbf{x})$
return $\bar{f}_l^*$, $\text{covar}(\bar{f}_l{}^*)$, $\mathbf{x}$, K_l

[2] In general this is a vector of weights, dependent upon the dimension of your observables

3.2 Augmented-input model

Compared to the single-input model, the augmented-input model is algorithmically very similar, see algorithm 2. However, the prior functions f are no longer dependent on just the original observations, they are now also dependent on the outputs of each layer, the posterior mean. The inspiration behind this approach has come from the findings of Duvenaud et al. [6], who propose a method to avoid developing flat structures in deep models. They refer to this flat structure as a pathology, and propose that in order to avoid pathologies, we must ensure that we have input-connected layers. That is, we ensure at each layer that we pass through both the previous output and the original observations. Duvenaud et al.[6] define a pathology in terms of the eigen-spectrum of the Jacobian, and state that in order to have a more dynamic data-space, you require a Jacobian with a spectrum composed of a number of large singular values after each layer. This implies with a high probability that the prior and posterior functions can evolve in more directions along our data manifold. The ability to move in multiple directions, in theory, enables us to model more dynamic behaviour. However, we find that although our covariance functions remain dynamic, our augmented-inputs model can become fixed in a particular state, that we call a *stale state*. This happens when the derivatives of the conditional joint posterior distribution with respect to the previous layer posterior mean, tends to zero.

We state, that due to the now augmented-inputs, the prior functions will be drawn from the following:

$$
\begin{aligned}
f_0 &\sim GP_0(\mathbf{0}, k_0(\mathbf{x}_0, \mathbf{x}_0)) \\
f_1 | \bar{f}_0^* &\sim \mathcal{GP}_1(\mathbf{0}, k_0(\mathbf{x}_1, \mathbf{x}_1) + k_1(\mathbf{x}_1, \mathbf{x}_1)) \\
&\vdots \\
f_{i+1} | \bar{f^*}_i &\sim \mathcal{GP}_i(\mathbf{0}, K_i^A = \sum_{l=0}^{N=i} k_l(\mathbf{x}_i, \mathbf{x}_i))
\end{aligned}
$$

where $\mathbf{x}_i$ represents the augmented i'th layer input. In the initial layer, $\mathbf{x}_0$, just represents our observables. In all succeeding layers $\mathbf{x}_i \in \mathbb{R}^{n \times (D+1)}$ represents the observations $\mathbf{x}$, augmented with the previous layer output $\bar{f^*}_{i-1}$, see algorithm 2 for more information. Notionally, not much has changed. However, we are now conditioning our next layer prediction on the previous layer output. Thus, the posterior mean for all layers after level 0, is given as:

$$
\begin{aligned}
\bar{f^*}_l &= \mathbb{E}[f_{l-1}^* | \mathbf{x}, \mathbf{y}_l, x_l^*, \bar{f^*}_{l-1}, K_l^A] \\
&= K_l^A(x_l^*, \mathbf{x}_{l-1})[K_l^A(\mathbf{x}_{l-1}, \mathbf{x}_l) + \sigma_y^2 \mathbb{I}]^{-1} \mathbf{y}_l
\end{aligned} \tag{14}
$$

There are some subtle points to note here, which are hidden amongst the indices. First, note that the test inputs are subscripted. This is because, after layer 0, the input data has an additional input dimension. Thus, to create the new test inputs we augment the original test inputs x^* with the full prediction made by the GP on all of the data. Second, the targets are subscripted, just as in the single-input case, to emphersise our assumptions about how $\mathbf{y}$ is distributed in each layer. In the augmented-input case, this is given by $\mathbf{y}_i \sim \mathcal{N}(\mathbf{0}, \sum_{l=0}^{N=i} k_l(\mathbf{x}_i, \mathbf{x}_i) + \sigma_y^2 \mathbb{I})$.

As stated previously, one disadvantage of this model is when we get stuck in a stale state. This happens when the outputs and inputs become very correlated, which leads to our Gaussian processes becoming over confident in its prediction. In order to avoid this stale state, our posterior distribution must continue to change in structure, so that the following derivative is never zero $\frac{\partial \mathcal{N}(\bar{f^*}_l, cov \bar{f^*}_l)}{\partial \bar{f^*}_{l-1}}$. Because we see that in the predictive posterior distribution, $\mathcal{N}(\bar{f^*}_l, cov \bar{f^*}_l)$, that not only do the covariances tend to zero, but so to do the variances along the diagonals. This means that when we sample from this distribution, the mass of the probability density is heavily focused on the mean. Therefore, as the number of layers increase, we will continue to draw the same value, which means there is no more randomness as our distribution tends to a point, the value of the output of the previous layer. Thus, in order to have a dynamic derivative, we need to ensure that the probability density of each test point remains well distributed. In a GP, this ultimately lies with the choice of kernel and hyperparameters. This is alluded to by Duvenaud et al.[6] and its importance should not be underestimated. Especially, if we want model rich structures.

Algorithm 2 Augmented-input stacked GPs regression

Input: $\mathbf{x}_0, x_0^*, \mathbf{y}_0, k_0, \theta, N, \sigma_y^2, i$
for $l = 0 : i$ **do**
  if $l > 0$ **then**
    $\mathbf{x}_l \leftarrow \bar{f}_{l-1}^*, \mathbf{x}_0$ ▷ the original observed values are augmented with the previous posterior
    $K_l^A \leftarrow K_{l-1} + k_{new}$ ▷ k_{new} is a new kernel selected at random
  else
    $K_l \leftarrow k_0$
  $L \leftarrow \text{cholesky}(K_l^A(\mathbf{x}_l, \mathbf{x}_l) + \sigma_y^2)$
  $\alpha \leftarrow L^T(L \backslash \mathbf{y}_l)$
  $\bar{f}_l^* \leftarrow K_l^A(x_l^*, \mathbf{x}_l)\alpha$ ▷ Posterior mean
  $\gamma \leftarrow L \backslash K_l^A(\mathbf{x}_l, x_l^*)$
  $\text{covar}(\bar{f}_l^*) \leftarrow K_l^A(x_l^*, x_l^*) - \gamma^T\gamma$ ▷ Posterior variance
  $\log P(\mathbf{y}_l|\mathbf{x}_l) \leftarrow -\frac{1}{2}\mathbf{y}_l^T\alpha - Tr(\log L) - \frac{n}{2}\log 2\pi$ ▷ Gaussian likelihood
  $\theta_l \leftarrow optimise\ \log P(\mathbf{y}_l|\mathbf{x}_l)$
return $\bar{f}_l^*$, $\text{covar}(\bar{f}_l{}^*)$, $\mathbf{x}_l$, K_l

4 Results

In this section we present a sub-selection of our results for both the single-input and augmented-input models[3]. We use a variety of data sets; stock-prices, sunspots [14], Weierstrass functions, Mackey-glass and heartbeat data [15]. These data sets all have interesting structures, and we apply our models to them to demonstrate both the strengths and weaknesses of the model. In general we do two tests, one using 20% of the data set and another using 70% of the data set, and then predict on the remaining data. Typically, we are more interested in how this model works on smaller amounts of data. In order to compare results between different models, we calculate the normalised root mean squared error (NRMSE) for all predictions made. We use the definition $NSRME = \frac{RMSE}{max(\mathbf{y}_{true}) - min(\mathbf{y}_{true})}$ to calculate the NSRME. Due to the architectures of both models, layer 0, will always represent a vanilla GP.

4.1 Weierstrass data set

The Weierstrass data set generated is from a summation of the third and fifth modes of a Weierstrass function, plus some Gaussian noise. This data set is quite tricky for a vanilla GP to predict as it is incredibly volatile. For both models, there is a choice at each layer between either a SE kernel, equation (8), or a standard periodic kernel, equation (9) . Kernel hyperparameters are randomised for both models[4]. To see how the structure in data is discovered, we show in figure 1 for the single-input model, and in figure 2, for the augmented-input model, how the model changes over five layers. What is quite interesting, is that even though at each layer both models selected the same kernels, we have very different behaviour. In the single-input case, what we see in layer 4 is precisely what we see in every layer before, except layer 0. What develops is a pathology, as the posterior mean remains flat in its prediction. However, in the augmented case, our model passes through information gained from the previous outputs. In this instance, it enables us to produce a much better prediction, despite having only 20% of the data set to predict on. In addition to this, we still have a well distributed posterior covariance, which enables us to model the uncertainty as well. This isn't always the case with the augmented-input model, see figure 3.

4.2 Heartbeat data

The heartbeat time-series data is based on 950 evenly-spaced measurements of instantaneous heart rate from a single subject performing a physical activity. The measurements (in units of beats per minute) occur at 0.5 second intervals, in our scale 0.5 seconds $\equiv \frac{1}{950}$, so that the length of the series is exactly 7 minutes and 55 seconds. For the augmented case we show the output at various layers,

[3] Datasets, results and code can be found on github.com/Bjgh/

[4] We will include the relevant .csv files that store the parameters and kernels chosen, for reproducibility.

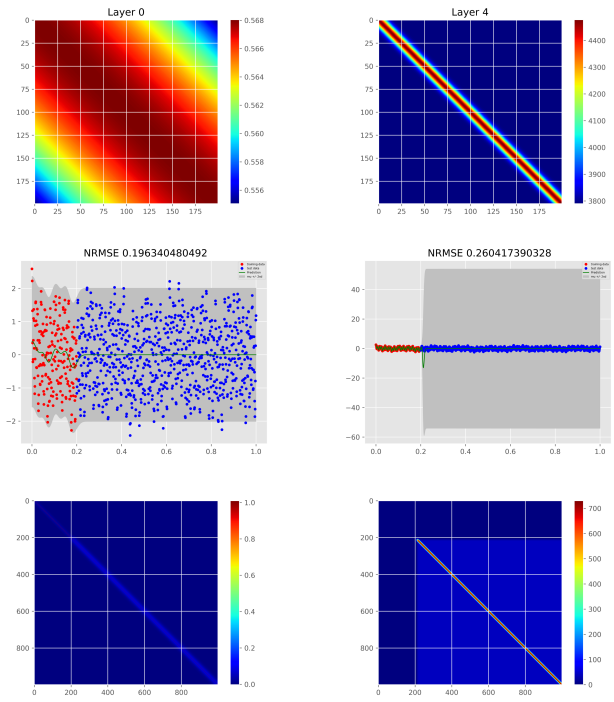

Figure 1: A prediction of the Weierstrass on 20% of the data set, 200 pts, in the single-input model. The top row is a linear combination of covariance functions operating on the original observations. The middle row represents the prediction, the posterior mean is in green, the observables are in red and the targets are in blue. The bottom row is the posterior covariance at the levels 0 and 4.

see figure 3. At each layer the model can select either the SE or standard periodic kernel. Kernel hyperparameters are chosen at random. It would be correct to say that this type of time series is not suited towards GPs. However, we show that we are able to model the more volatile aspects of the data, such as the six strong peaks, that a vanilla GP would typically struggle with, see layer 2 in figure 3. Equally, with this data set we are able to highlight a particular problem with the augmented-input method, that being minimal change in the posterior means from layer to layer, cause the model to become over confident in itself and get stuck in a particular state, the stale state, as seen in layer 4 in figure 3. Despite how the kernel changes, the prediction remains the same. Empirically, we find that in this dataset and others, when this stale state occurs, we have two problems. One, the GP produces a posterior covariance of values all equal to near-zero as the GP prediction becomes increasingly sure of itself. This of course dampens our efforts to model uncertainty. Two, the entries in the covariance function tend to very large values. This means that the hyperparameters, in particular the vertical scale variance hyperparameters, are also tending to large values. As the posterior mean is no longer dynamic, and as many of our kernels are part of the exponential family, we see that the distance metrics within the kernels will produce many zeros. Thus, the exponentials will be equal to one in many locations. Therefore, it is the vertical scale variance hyperparameters that will dictate the structure of the covariance function. To avoid such problems you can constrain the hyperparameters, or choose kernels that are outside the exponential family.

4.3 Stock data

This data set is comprised of 20 different stocks and contains 5000 days worth of opening and closing values, of particular companies. We only use 1000 days worth of data, 1994 to 1997. Again, as the data itself is highly volatile, it makes it difficult to predict future points using a vanilla GP. For this data set, each model can choose from either the SE, standard periodic or Matern $\frac{3}{2}$ ($p = 1$) kernel. The kernel hyperparameters are randomised. In running the single-input model on this data set, we find that we get an enhanced prediction, as compared to a vanilla GP, see figure 4. We see that in layer 0, the GP layer, a Matern $\frac{3}{2}$ kernel is selected and it manages to model the next 11 time points

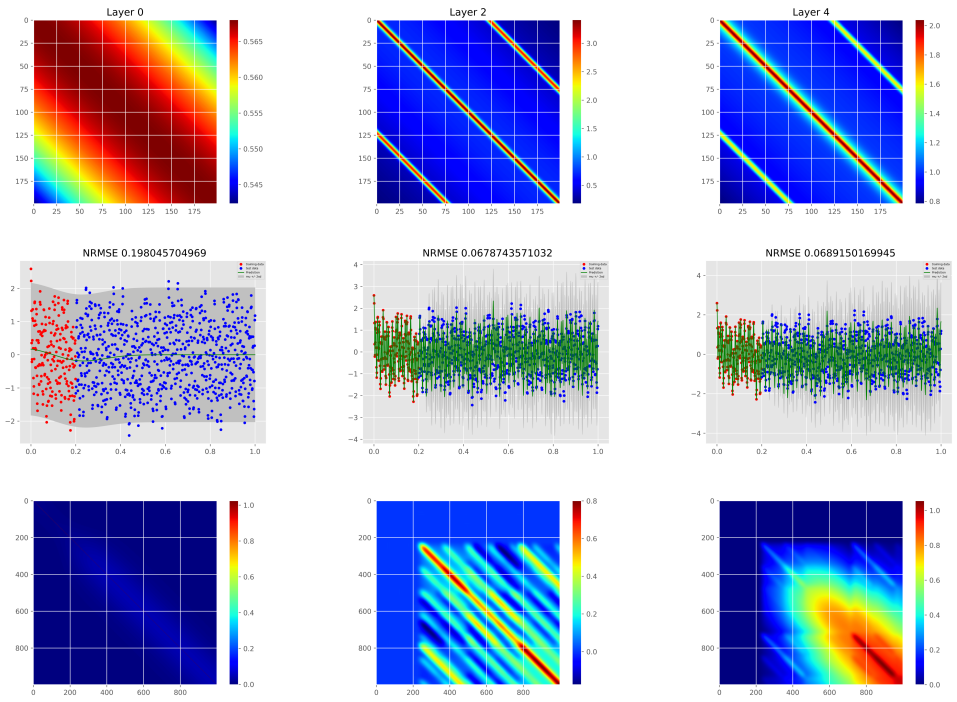

Figure 2: A prediction of the Weierstrass on 20% of the data set, 200 pts, in the augmented-input model. The top row is a linear combination of covariance functions operating on the original observations. The middle row represents the prediction, the posterior mean is in green, the observables are in red and the targets are in blue. The bottom row is the posterior covariance at the levels 0, 2 and 4.

in the future relatively well. Then it receives a boost in layer 2 from a SE kernel, which directs it on a upward trend. But, keeps the prediction smooth. In layer 3, the model selects a Matern $p = \frac{3}{2}$ kernel again, which adds a a little volatility to the prediction. It then selects four standard periodic kernels, which amplify this effect. Creating this kind of structure, given the already seen data, with a vanilla GP would be difficult. But, due to the additive nature of the kernels in the single-input model, we are able to create these structures.

4.4 Mackay-glass data

The Mackay-Glass equation is a non-linear time delay differential equation, that can generate complex dynamics and is particularly useful in physiological feedback systems. A vanilla GP, with a standard periodic kernel and the right hyperparameters, for example $h = 0.6$ and $l = 0.1$, will give you a relatively good prediction. However, with the augmented-inputs model we are able to generate not only a good prediction, but some very interesting behaviour, see figure 5. At each layer we only allow the model to select a standard periodic kernel, with randomised kernel hyperparameters. Initially, in layer 0, we generate a flat prediction, this is due to having selected a large lengthscale l. However, in layer 1 we begin to generate more interesting structure, which leads us to some rather odd, but dynamic structure in layer 2. From the covariance function, you can see that whilst the values remain low, which is required for a dynamic posterior mean, we have this beautiful structure. We then progress further, and find that at layer 4 the model is starting to get the right phase and structure of the data. This leads to a fascinating covariance function. Where, we have coming off the diagonal, numerous small and well distributed values, which is difficult to see from afar.

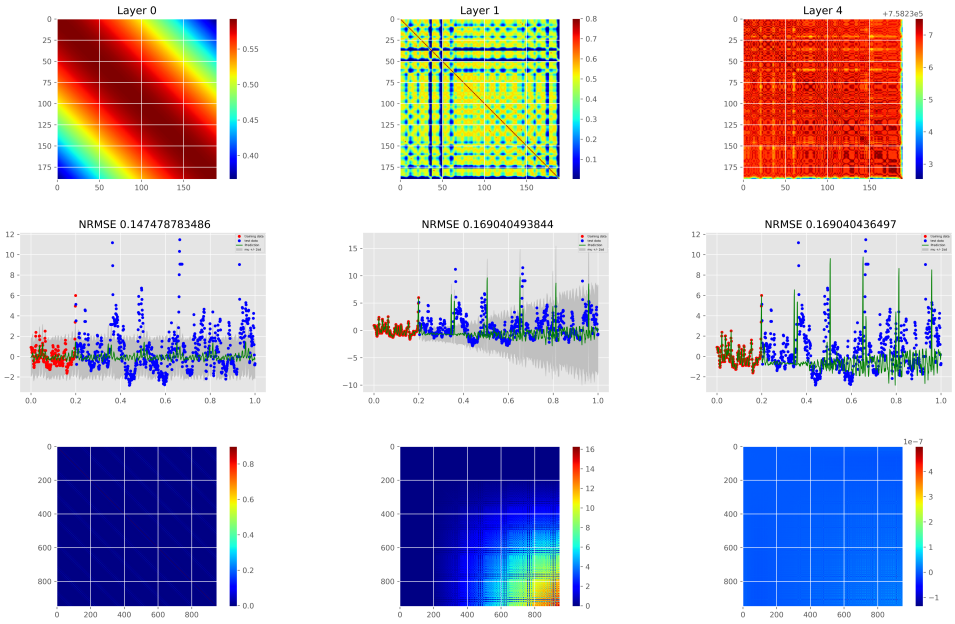

Figure 3: A prediction of the heartbeat time series on 20% of the data set, 190 pts, in the augmented-input model. The top row is a linear combination of covariance functions operating on the original observations. The middle row represents the prediction, the posterior mean is in green, the observables are in red and the targets are in blue. The bottom row is the posterior covariance at the levels 0, 1 and 4. The negative values in the posterior covariance are produced by numerical errors.

5 Conclusions

In this work we have presented two models, the single-input and the augmented-input model. We have shown that both models can model real world data using a stacked kernel approach and can outperform that of a vanilla Gaussian process. We have shown empirically that both models can find underlying structure in the data, even for small data sets. We have shown that for our augmented model, the theorems prescribed in Duvenaud et al. [6] do not necessarily hold for the augmented-input model that we have designed. We have discussed how we may avoid stale states from occurring. For future work, we aim to derive analytically a more robust approach for avoiding stale states.

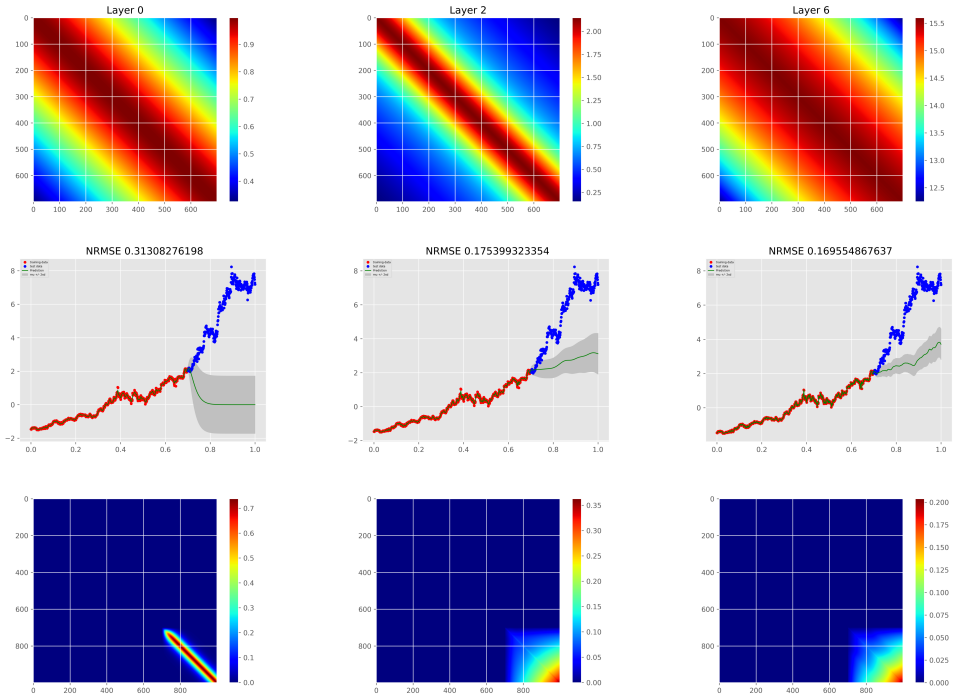

Figure 4: A prediction of Merck & Co (ticker: MRK) opening stock price on 70% of the data set, 700 pts, in the single-input model. The top row is a linear combination of covariance functions operating on the original observations. The middle row represents the prediction, the posterior mean is in green, the observables are in red and the targets are in blue. The bottom row is the posterior covariance at the levels 0, 2 and 6.

References

[1] S. Ghoshal and S. Roberts, "Extracting predictive information from heterogeneous data streams using Gaussian Processes," *Algorithmic Finance*, vol. 5, no. 1-2, pp. 21–30, 2016.

[2] S. Bratieres, N. Quadrianto, and Z. Ghahramani, "Bayesian structured prediction using Gaussian Processes," *arXiv preprint arXiv:1307.3846*, 2013.

[3] V. Rajpaul, S. Aigrain, M. A. Osborne, S. Reece, and S. Roberts, "A Gaussian Process framework for modelling stellar activity signals in radial velocity data," *Monthly Notices of the Royal Astronomical Society*, vol. 452, no. 3, pp. 2269–2291, 2015.

[4] L. Clifton, D. A. Clifton, M. A. Pimentel, P. J. Watkinson, and L. Tarassenko, "Gaussian Processes for personalized e-health monitoring with wearable sensors," *IEEE Transactions on Biomedical Engineering*, vol. 60, no. 1, pp. 193–197, 2013.

[5] G. S. Aoude, B. D. Luders, J. M. Joseph, N. Roy, and J. P. How, "Probabilistically safe motion planning to avoid dynamic obstacles with uncertain motion patterns," *Autonomous Robots*, vol. 35, no. 1, pp. 51–76, 2013.

[6] D. Duvenaud, O. Rippel, R. Adams, and Z. Ghahramani, "Avoiding pathologies in very deep networks," in *Artificial Intelligence and Statistics*, pp. 202–210, 2014.

[7] R. M. Neal, *Bayesian learning for Neural Networks*. PhD thesis, University of Toronto, 1995.

[8] A. Damianou and N. Lawrence, "Deep Gaussian Processes," in *Artificial Intelligence and Statistics*, pp. 207–215, 2013.

[9] C. K. Williams and C. E. Rasmussen, "Gaussian Processes for machine learning," *the MIT Press*, vol. 2, no. 3, p. 4, 2006.

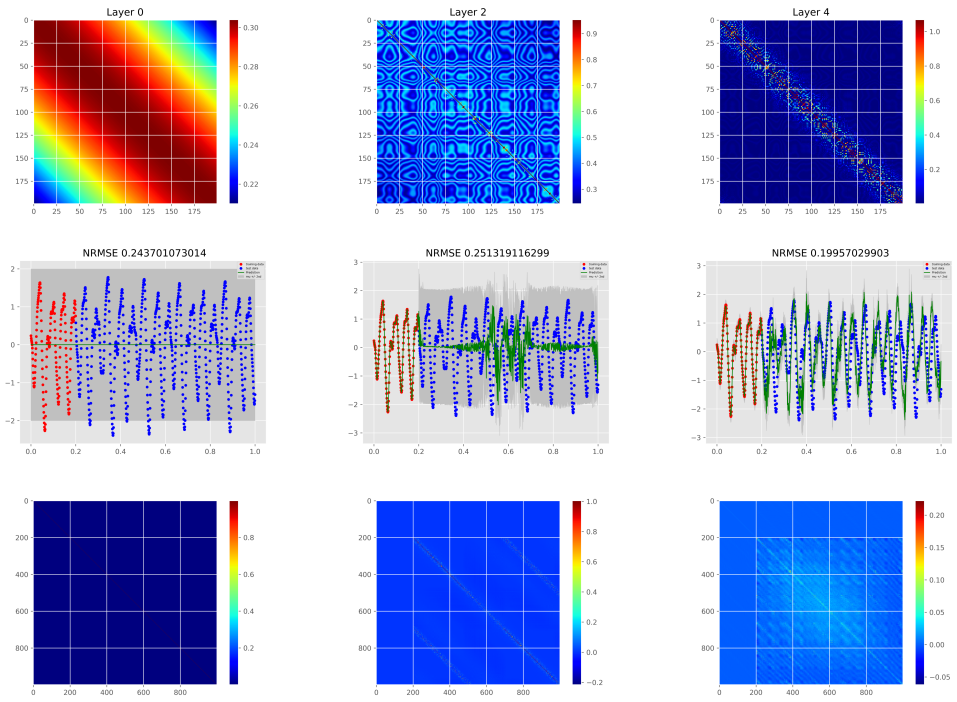

Figure 5: A prediction on 20% of the Mackay-glass data, 200 pts, in the augmented-input model. The top row is a linear combination of covariance functions operating on the original observations. The middle row represents the prediction, the posterior mean is in green, the observables are in red and the targets are in blue. The bottom row is the posterior covariance at the levels 0, 2 and 4. The negative values in the posterior covariance are produced by numerical errors.

[10] D. J. MacKay, "Introduction to Gaussian Processes," *NATO ASI Series F Computer and Systems Sciences*, vol. 168, pp. 133–166, 1998.

[11] C. K. Williams, "Computing with infinite networks," *Advances in neural information processing systems*, pp. 295–301, 1997.

[12] GPy, "GPy: A Gaussian Process framework in python." `http://github.com/SheffieldML/GPy`, since 2012.

[13] D. K. Duvenaud, H. Nickisch, and C. E. Rasmussen, "Additive Gaussian Processes," in *Advances in neural information processing systems*, pp. 226–234, 2011.

[14] SILSO World Data Center, "The International Sunspot Number," *International Sunspot Number Monthly Bulletin and online catalogue*.

[15] A. L. Goldberger, L. A. N. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley, "Physiobank, physiotoolkit, and physionet: Components of a new research resource for complex physiologic signals," *Circulation*, vol. 101, no. 23, pp. e215–e220, 2000 (June 13). Circulation Electronic Pages: http://circ.ahajournals.org/content/101/23/e215.full PMID:1085218; doi: 10.1161/01.CIR.101.23.e215".