1 Gaussian Process

[1] introduces Gaussian Process regression as an alternative view to Bayesian regression.

1.1 Bayesian Regression

In linear regression setup, we assume output $y \in \mathbb{R}$ is a linear function of inputs $x \in \mathbb{R}^d$, corrupted with additive iid normal noise,

$$y = f(x) + \epsilon \quad \text{where} \quad f(x) = w^T x \quad \text{and} \quad \epsilon \sim \mathcal{N}(0, \sigma_n^2) \tag{1}$$

The Bayesian setup considers $w \in \mathbb{R}^d$ as a random variable, endowed with prior $w \sim \mathcal{N}(0, \Sigma_p)$. Using Bayes rule, we can find the posterior of weights given data, which is again a normal random variable $p(w \mid X, y) = \mathcal{N}\left(w\,; A^{-1}b, A^{-1}\right)$ where $A = \frac{1}{\sigma_n^2} X^T X + \Sigma_p^{-1}$ and $b = \frac{1}{\sigma_n^2} X^T y$ and $X \in \mathbb{R}^{n \times d}, y \in \mathbb{R}^{n \times 1}$ are design matrices. For test point x_*, the predictive distribution of $f_* = f(x_*)$ is the average likelihood of f_* under model $f(x; w)$ with respect to posterior of w.

$$p(f_* \mid x_*, X, y) = \int p(f_* \mid x_*, w) p(w \mid X, y)\, dw \tag{2}$$

We can think of the predictive distribution as a linear function $f_* = x_*^T w$ of weights, a normal random variable, and therefore is normal. Therefore, $f_* \mid x_*, X, y \sim \mathcal{N}(x_*^T A^{-1} b, x_*^T A^{-1} x_*)$. The natural extension to Bayesian linear regression is to kernelize it, for example assume a linear model in some feature space $f(x) = \phi(x)^T w$ for some feature map $\phi : \mathbb{R}^d \to \mathbb{R}^D$. We can write predictive distribution as

$$f_* \mid x_*, X, y \sim \mathcal{N}(k(x_*, X)(k(X, X) + \sigma_n^2 I)^{-1} y, \tag{3}$$

$$k(x_*, x_*) - k(x_*, X)(k(X, X) + \sigma_n^2 I)^{-1} k(X, x_*)) \tag{4}$$

where $k(X, X') = \Phi \Sigma_p \Phi'^T \in \mathbb{R}^{n \times n'}$ and $\Phi \in \mathbb{R}^{n \times D}$ are the feature vectors.

1.2 Gaussian Process

Definition 1. *(Gaussian Process) A Gaussian process is a stochastic process $\{X_t\}_{t \in T}$ such that, for every finite subset of indices $t_1, \cdots, t_k \in T$, $(X_{t_1}, \cdots, X_{t_k})$ is multivariate normal.*

A Gaussian process $f \sim \mathcal{GP}(m, k)$ over $\mathbb{R}^{\mathcal{X}}$ is fully specified by the mean function $m : \mathcal{X} \to \mathbb{R}$ and covariance function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ where

$$m(x) = \mathbb{E}\left[f(x)\right] \qquad k(x, x') = \mathbb{E}\left[(f(x) - m(x))(f(x') - m(x'))\right] \tag{5}$$

The covariance function determines function's behavior, for example its stationarity, smoothness, and periodicity etc. For example, the squared exponential covariance function $k_{\text{SE}}(x, x') = \exp(-\frac{1}{2\ell^2} \|x - x'\|_2^2)$ enforces the prior knowledge that functions are smooth, i.e. inputs are close in the Euclidean sense will have similar outputs. See Figure (1) for some examples of samples from Gaussian process.

Example 1. *(Intuition about covariance matrix for $\mathcal{GP}$) First consider $(y_1, y_2) \sim \mathcal{N}(\mathbf{0}, \Sigma)$ where $\Sigma_{11} = \Sigma_{22} = 1$ and $\Sigma_{12} = \Sigma_{21} = \rho_{12}$. We know $y_1 \mid y_2 = a \sim \mathcal{N}(\rho_{12} a, 1 - \rho_{12}^2)$. When $Cov(y_1, y_2) = \rho_{12} \uparrow 1$, y_1's samples conditioned on $y_2 = a$ fall close to a with high probability. When $\rho_{12} \downarrow 0$, $y_1 \mid y_2 = a$ will distribute like a unit normal, regardless of values that y_2 take. Extend this intuition to gaussian process: whenever $k(x_i, x_j)$ is large, y_i, y_j are correlated and so observing $y_i = a$ provides a strong prior on how y_j will behave, or in other words, reduce the uncertainty of values that y_j can take dramatically.*

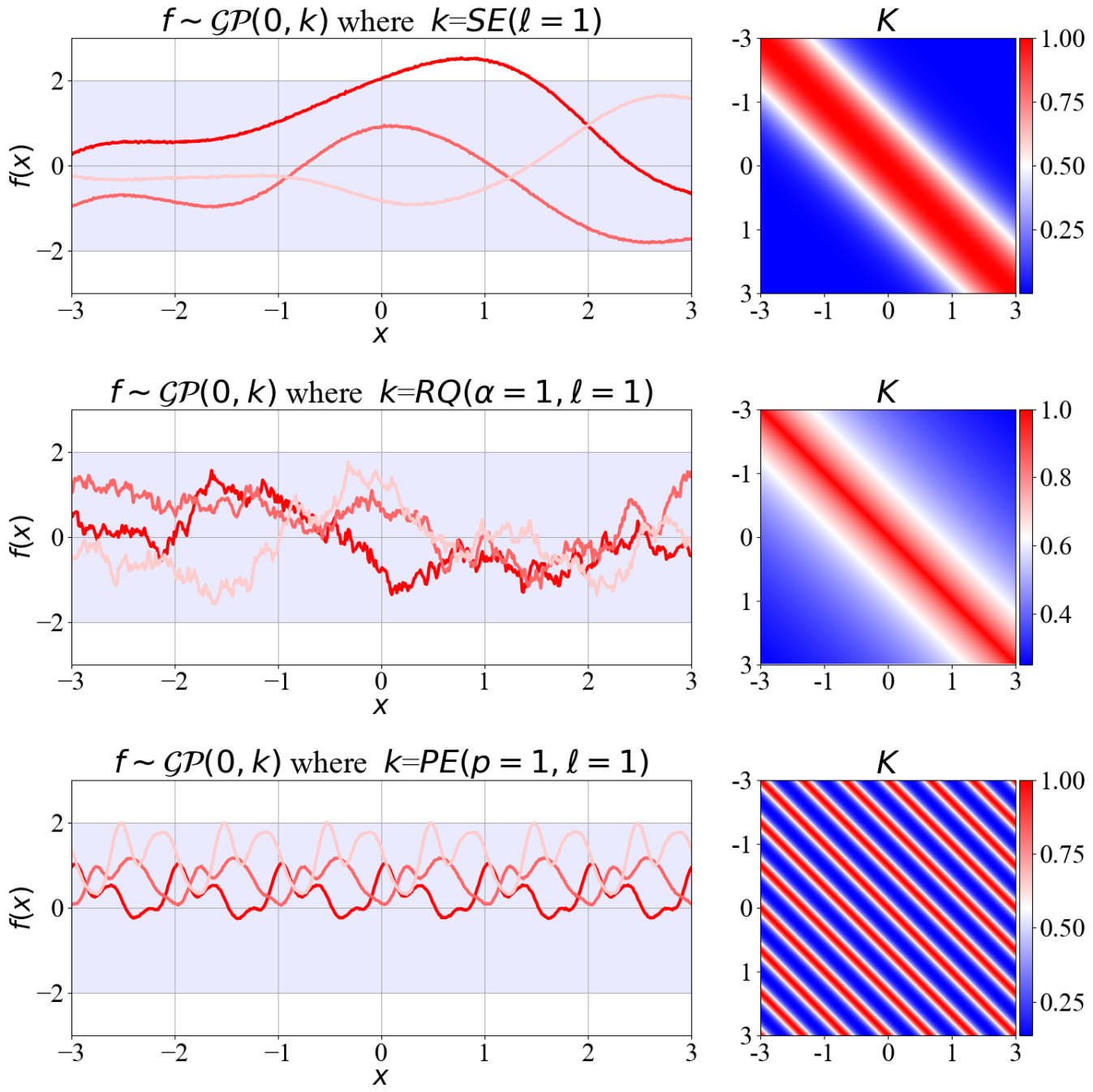

Figure 1: (Left) Samples from Gaussian process prior and (Right) covariance matrix at test locations.

1.3 Gaussian Process Regression

Instead of placing a prior over weights $p(w)$ to quantify randomness in function $f(x) = \phi(x)^T w$, we model function directly as a Gaussian process, $f \sim \mathcal{GP}(0, k)$. There is a one-to-one correspondence between the two views. For example, $f(x) = \phi(x)^T w$ with prior $w \sim \mathcal{N}(0, \Sigma_p)$ used in kernel Bayesian regression has

$$\mathbb{E}\left[f(x)\right] = \phi(x)^T \mathbb{E}\left[w\right] = 0 \qquad \mathbb{E}\left[f(x)f(x')\right] = \phi(x)^T \Sigma_p \phi(x') \tag{6}$$

Therefore, $f \sim \mathcal{GP}(0, k)$ where $k(x, x') = \phi(x)^T \Sigma_p \phi(x')$. Note k is in fact a valid kernel. (Since Σ_p is psd, $\Sigma_p = UDU^T$ by SVD. We can write $k(x, x') = \langle \psi(x), \psi(x') \rangle$ where $\psi(x) = \Sigma_p^{1/2} \phi(x)$ and $\Sigma_p^{1/2} = UD^{1/2}U^T$). Conversely, any valid kernel used in kernel Bayesian regression can be used to parameterize the covariance function of the Gaussian process model over f.

Since $y = f(x) + \epsilon$, we have $\mathbf{y} = \mathbf{f} + \sigma_n^2 I \sim \mathcal{N}(0, k(X, X) + \sigma_n^2 I)$ where $\mathbf{y}, \mathbf{f} \in \mathbb{R}^{n \times 1}$. We can write the joint distribution of observed values and function values at some test locations X_* as

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} k(X, X) + \sigma_n^2 I & k(X, X_*) \\ k(X_*, X) & k(X_*, X_*) \end{bmatrix}\right) \tag{7}$$

We can derive the predictive distribution for $\mathbf{f}_* \mid \mathbf{y}$ by simply apply conditional distribution formula

$$\mathbf{f}_* \mid X, \mathbf{y}, X_* \sim \mathcal{N}(k(X_*, X)(k(X, X) + \sigma_n^2 I)^{-1} \mathbf{y} \tag{8}$$

$$k(X_*, X_*) - k(X_*, X)(k(X, X) + \sigma_n^2 I)^{-1} k(X, X_*)) \tag{9}$$

which has exact form compared to (4). More compactly, for a single test point x_*, $\mu_{\mathbf{f}_*} = k_*^T (K + \sigma_n^2 I)^{-1}$ and $\text{Var}(\mathbf{f}_*) = k(x_*, x_*) - k_*^T (K + \sigma_n^2 I)^{-1} k_*$ where $k_* = k(X, x_*) \in \mathbb{R}^{n \times 1}$. See Figure (2) for examples of Gaussian process regression with varying data size and hyperparameters.

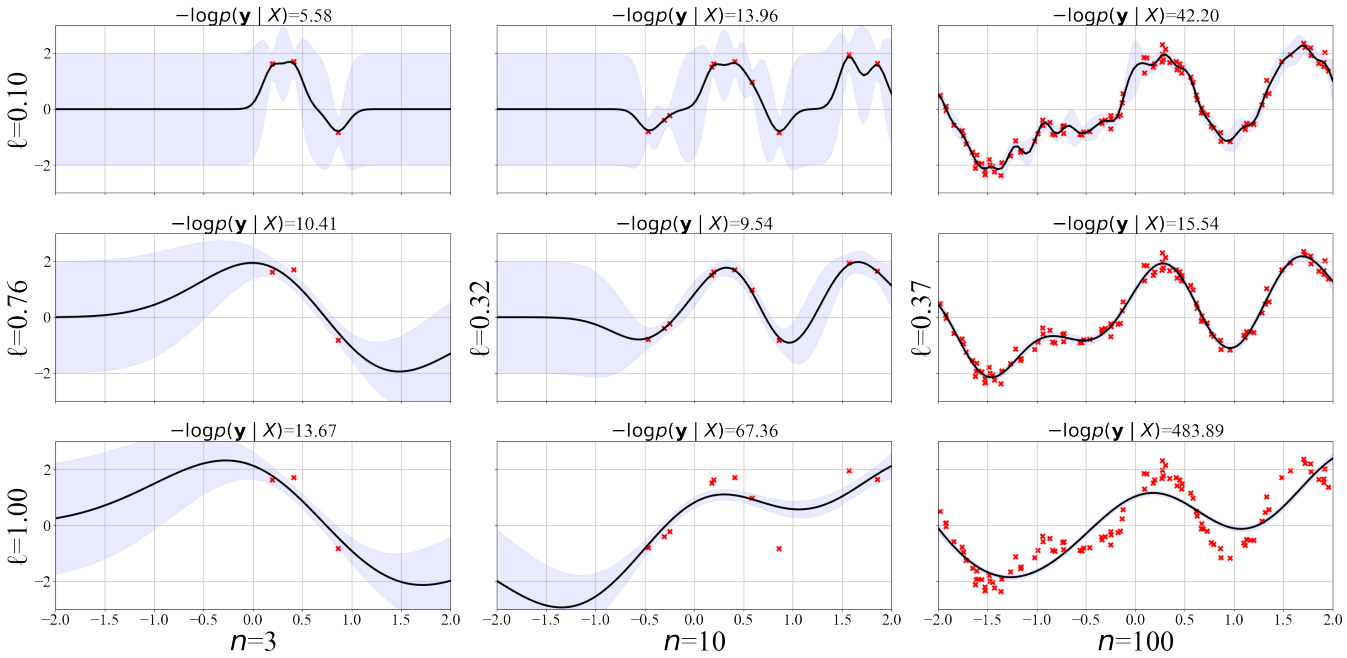

Figure 2: This plot shows mean (black) and 95% confidence interval (light blue) for predictive distribution $p(\mathbf{f}_* \mid X, \mathbf{y}, X_*)$ fit using Gaussian process regression assuming a SE prior over $f \sim \mathcal{GP}(0, k_{SE})$ of varying lengthscale ℓ and observed sample sizes n given groundtruth noise $\sigma_n = 0.1$. The middle row's hyperparameters is taken to be the empirical Bayes estimate $\ell, \sigma_n = \arg\max p(\mathbf{y} \mid X, \ell, \sigma_n)$ where $p(\mathbf{y} \mid X, \ell, \sigma_n)$ given by Equation (11), optimized via gradient descent. Note estimating $\log(\ell), \log(\sigma_n)$ makes the optimization problem much easier.

1.4 Model Selection

Model selection for Gaussian process regression involves picking the form and the hyperparameters of the covariance function. Given data $(X, \mathbf{y})$, marginal likelihood $p(\mathbf{y} \mid X)$ quantifies how likely data is observed under our additive noise model $\mathbf{y} \mid \mathbf{f} \sim \mathcal{N}(\mathbf{f}, \sigma_n^2 I)$ on average with respect to latent function values (or parameters of our model) $\mathbf{f} \mid X \sim \mathcal{N}(0, K)$ (due to assumption of Gaussian process prior over f),

$$p(\mathbf{y} \mid X) = \int p(\mathbf{y} \mid \mathbf{f}, X) p(\mathbf{f} \mid X) \, d\mathbf{f} \tag{10}$$

This formulation is analogous to that in Bayesian linear regression where the marginal likelihood marginalizes over weights $p(\mathbf{y} \mid X) = \int p(\mathbf{y} \mid X, w) p(w) \, dw$. We can obtain a closed form expression by reading off (7), i.e. $\mathbf{y} \sim \mathcal{N}(\mathbf{0}, K + \sigma_n^2 I)$.

$$\log p(\mathbf{y} \mid X) = -\frac{1}{2} \mathbf{y}^T (K + \sigma_n^2 I)^{-1} \mathbf{y} - \frac{1}{2} \log |K + \sigma_n^2 I| - \frac{n}{2} \log 2\pi \tag{11}$$

In Bayesian model selection, model hyperparameters for Gaussian process regression $\theta = \{\sigma_n, \ell\}$ can be found by maximizing the marginal likelihood or the type II likelihood $\theta^* = \arg\max_\theta \log p(\mathbf{y} \mid X, \theta)$.

2 Multitask Learning

2.1 MTGP

Multitask Gaussian Process (MTGP) regression [2] is a method to do multitask learning using Gaussian process. Given design $X \in \mathbb{R}^{N \times D}, Y \in \mathbb{R}^{N \times M}$, we want to learn a vector valued regressor $f : \mathbb{R}^D \to \mathbb{R}^M$ that fits data well. We can put a GP prior over $f \sim \mathcal{GP}(0, k)$ where the covariance function models both the relationship between inputs via k_x and similarity between tasks / output coordinate via k_t, specifically define k as tensor product $k_x \otimes k_t$ over $\mathcal{X} \times \mathcal{T}$ where $\mathcal{T} = \{1, 2, \cdots, M\}$ is space of tasks,

$$k((x, t), (x', t')) = k_x(x, x') \, k_t(t, t') \tag{12}$$

Consider a likelihood model with task specific noise variance $y_t \sim \mathcal{N}(f_t(x), \sigma_t^2)$. Denote Y_t as t-th column in Y, then $Y_t \sim \mathcal{N}(0, k_t(t,t)K + \sigma_t^2 I)$ where $[K]_{ij} = k_x(x_i, x_j)$. Furthermore, if we let $\mathbf{y} = \text{vec}(Y) = (Y_1, \cdots, Y_M)$, then $\mathbf{y} \sim \mathcal{N}(0, \Sigma)$ where

$$\Sigma = K^t \otimes K + D \otimes I = \begin{bmatrix} K_{11}^t K + \sigma_1^2 I & K_{12}^t K & \cdots & K_{1M}^t K \\ K_{21}^t K & K_{22}^t K + \sigma_2^2 I & \cdots & K_{2M}^t K \\ \vdots & \vdots & \ddots & \vdots \\ K_{M1}^t K & K_{M2}^t K & \cdots & K_{MM}^t K + \sigma_M^2 I \end{bmatrix} \tag{13}$$

Note this is the covariance matrix with no missing outputs, which leads to efficient inference methods. The task similarity matrix K_t ($[K_t]_{ij} = k_t(i, j)$) induce correlation between tasks. Intuitively, if two tasks t, t' are related in the sense an optimal regressor for $y_t, y_{t'}$ vary together in some systematic manner, then observation at some location (x, y_t) constrains what the value $y_{t'}$ can take. The degree to which we can reduce the uncertainty that $y_{t'}$ can takes depends on $k_t(t, t')$. Therefore, a learner which utilizes information obtained from related tasks can be learnt more efficiently than if learnt without knowledge of other tasks, in which case $K_t = I$. [2] proposes to use EM to first impute missing latent variables $\mathbf{f}$ from noisy observations $\mathbf{y}$, then find hyperparameters $\theta = \left\{\theta_{k_x}, K^t\right\}$ by maximizing the full data likelihoods $p(\mathbf{y}, \mathbf{f} \mid \theta_k)$. Alternatively, we can simply can learn the hyperparameters by applying gradient descent to maximize type-II likelihood $\theta^* = \arg\max_\theta \log p(\mathbf{y} \mid X, \theta)$. See Figure (3) for an example of MTGP in case where $D = 1$ and $M = 2$.

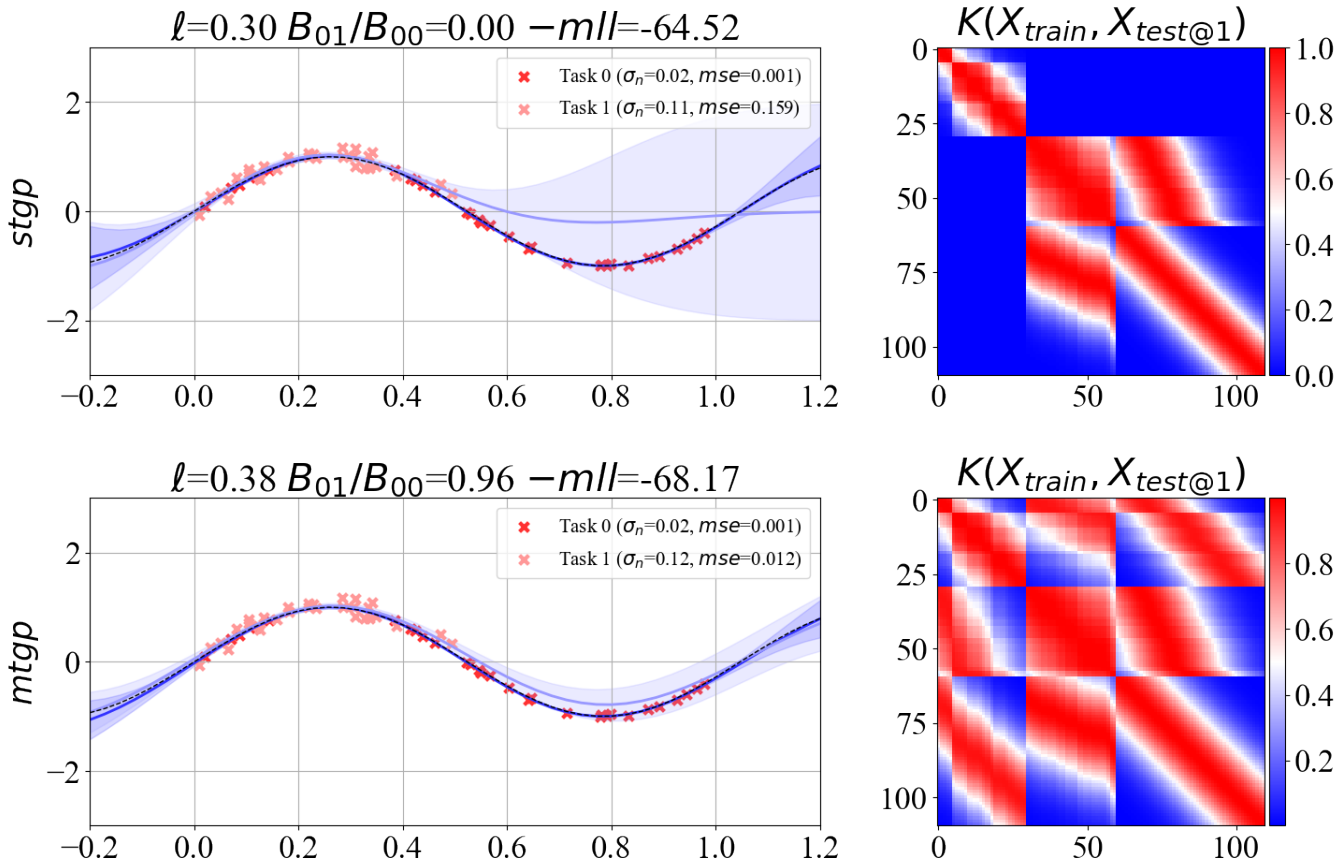

Figure 3: Observations for the two tasks are generated $y_0(x) = \sin(6x) + .03\epsilon$ and $y_1(x) = \sin(6x + \texttt{shift}) + .1\epsilon$ where $\epsilon \overset{iid}{\sim} \mathcal{N}(0, 1)$ and $\texttt{shift} = .3$. Here for task 1 we only sample points in $[0, .5]$ while for task 0 we sample points in $[0, 1]$. (Top) Two Gaussian process regressor are learnt to fit data independently. (Bottom) A MTGP is fit to data whereby the task similarity matrix $B := K^t$ is learnt by maximizing marginal log likelihood. We see that by taking into account of relationship between tasks, samples from task 0 reduces uncertainty that task 1 has at test locations far from its training support.

2.2 Asymmetric MTGP

When the goal is to improve performance of a target/main task given the other tasks, optimizing for maximum marginal likelihood of all tasks is suboptimal, as shown in Figure (4). There are extension of MTGP to the asymmetric case by assuming that auxiliary task functions are addition of a shared main task function and task specific function [3], however it still uses the full marginal likelihood for all tasks to find task similarity and kernel hyperparameters. Ideally, we want

1. An objective which depends on likelihood of main task only
2. Automatically learn the task similarity such that auxiliary task or side information can improve learning of main task. If a side task is totally unrelated to the main task, it should have minimal influence on learning of the main task.

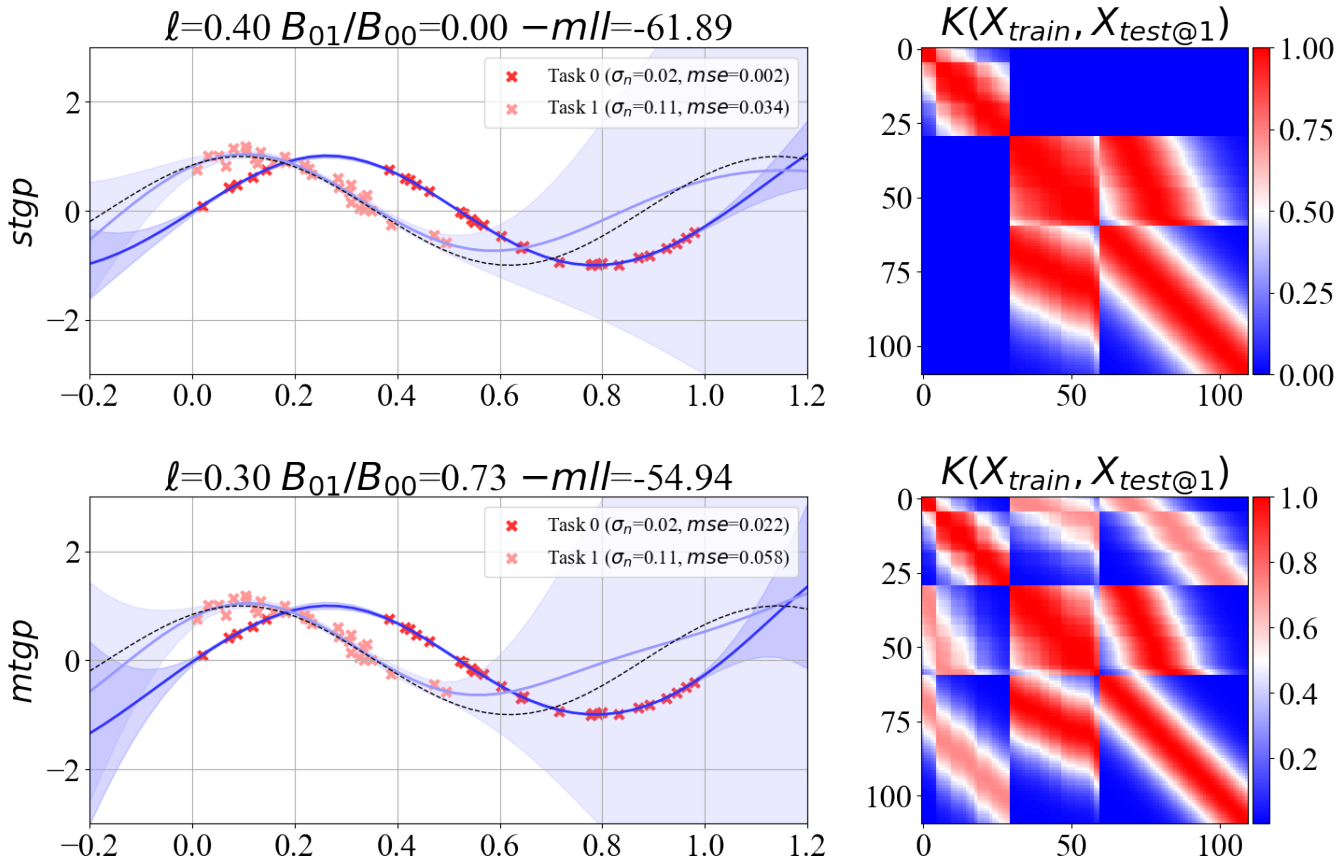

Figure 4: Observations for the two tasks are generated $y_0(x) = \sin(6x) + .03\epsilon$ and $y_1(x) = \sin(6x + \texttt{shift}) + .1\epsilon$ where $\epsilon \overset{iid}{\sim} \mathcal{N}(0, 1)$ and and $\texttt{shift} = 1.5$. Here for task 1 we only sample points in $[0, .5]$ while for task 0 we sample points in $[0, 1]$. There are 4 times training points for task 0 than task 1 (Top) Two Gaussian process regressor are learnt to fit data independently. (Bottom) A MTGP is fit to data whereby the task similarity matrix $B := K^t$ is learnt by maximizing marginal log likelihood. We see in case of data imbalance, which often happens in multitask learning, we arrive at an optimal marginal likelihood for all data points (a larger marginal likelihood), but renders prediction suboptimal for task 1 (a higher MSE)

Given design $X \in \mathbb{R}^{N \times D}, Y \in \mathbb{R}^{N \times 1}, S \in \mathbb{R}^{N \times M}$, we want to learn a scalar valued regressor $f : \mathbb{R}^D \to \mathbb{R}$ that fits the data well. We can alternatively try to learn the scalar valued regressor taking into account of relevant side information $f : \mathbb{R}^D \times \mathbb{R}^M \to \mathbb{R}$. Relevant side information that helps with learning the main task should be selected from potentially irrelevant side information. A simple thing one can do is to define a kernel that captures similarity between inputs via k_x and similarity of side information k_s, similar to the setup of MTGP. Let k be a tensor product $k_x \otimes k_s$ over $\mathcal{X} \times \mathcal{S}$,

$$k((x, s), (x', s')) = k_x(x, x')k_s(s, s') \tag{14}$$

Instead of task similarity matrix, correlation between any pair y, y' depends on value of $k_s(s, s')$. In the case where s describes a particular sub-population of our training dataset, it gives rise to a stratification dependent kernel. In health care domain, we might want to provide stratification dependent predictions given our side information of scanner, patient orientation, and existence of comorbidities. In this setting,

automatic selection of relevance can be achieved via an Automatic Relevance Determination (ARD) kernel

$$k_s(s, s') = \sigma^2 \exp\left(-\frac{1}{2}\sum_{l=1}^{M}\frac{1}{\ell_l^2}(s_l - s'_l)^2\right) \tag{15}$$

If ARD kernel is used, current formulation captures the prior that main task regressor is smooth with respect to some stratification of dataset - at test time the model will give similar prediction to those that appears to be likely in the same training stratification, as shown in Figure (7). Hyperparameters $\left\{\{\ell_l\}_{l=1}^{M}, \theta_{k_x}\right\}$ can be learnt by maximizing marginal likelihood.

One complication is missing side information during training, no information exists for some components of s. A more serious issue is at test time, we would want our parameterized function f to make prediction based on x_* only, absent of any side information. Naively this is not possible, since our regressor is supported over $\mathcal{X} \times \mathcal{S}$. A easy way to fix this is to pre-train a model with the goal of predicting side information from x - we first impute the missing side information, then proceed with both training and testing. A more principled way to do this is to train some GP model and use its predictive distribution over the side information as inputs for training and testing of f. This is related to GP with uncertain inputs and there is closed form solution for inference.

For a toy case where $D = 1, M = 1$, we see that the asymmetric MTGP described just now is able to utilize side information when they help Figure (5) and ignore side information when they are irrelevant Figure (6). A summary of model performance for main task regressor with varying side information relevance is in Table (1). A summary of how much `amtgp` depends on side information is in Table (2).

`shift`	`stgp`	`mtgp`	`amtgp`
0.000	0.579	0.021	0.181
0.100	0.660	0.030	0.106
0.200	0.735	0.026	0.088
0.300	0.802	0.027	0.162
0.400	0.857	0.041	0.246
0.500	0.898	0.070	0.256
1.000	0.837	0.139	0.296

Table 1: Mean squared error for main task (task=1) regressor at test locations along the grid (`linspace`(0,1,100)) for the three different methods `stgp`,`mtgp`,`amgtp` trained on data of varying task similarity. Larger values of `shift` implies the two task functions are more unrelated. `mtgp` is more flexible than `stgp` and therefore gives a higher marginal likelihood assuming the task similarity matrix can be found properly. `amtgp` is able to utilize side information when they are really relevant `shift` $= 0$, and does not suffer from performance drop when the two tasks are irrelevant `shift` $= 1.0$

shift	$\frac{\ell_s}{\ell_x}$
0.0	0.1
0.1	0.7
0.5	8.8
1.0	23.4
1.5	29.4

Table 2: Ratio of lengthscale for k_s over length scale for k_x, e.g. $\frac{\ell_s}{\ell_x}$ for main task (task=1) regressor trained via `amtgp`. A larger ratio implies the model relies more on x and less on side information s

References

[1] Carl Edward Rasmussen and Williams Christopher. *Gaussian Process for Machine Learning*. MIT Press, 2006.

[2] Edwin V Bonilla, Kian Ming A Chai, and Christopher K I Williams. "Multi-task Gaussian Process Prediction". In: (2008), p. 8.

[3] Gayle Leen, Jaakko Peltonen, and Samuel Kaski. "Focused multi-task learning in a Gaussian process framework". In: *Machine Learning* 89.1 (Oct. 1, 2012), pp. 157–182. ISSN: 1573-0565. DOI: 10.1007/s10994-012-5302-y. URL: https://doi.org/10.1007/s10994-012-5302-y (visited on 02/15/2021).

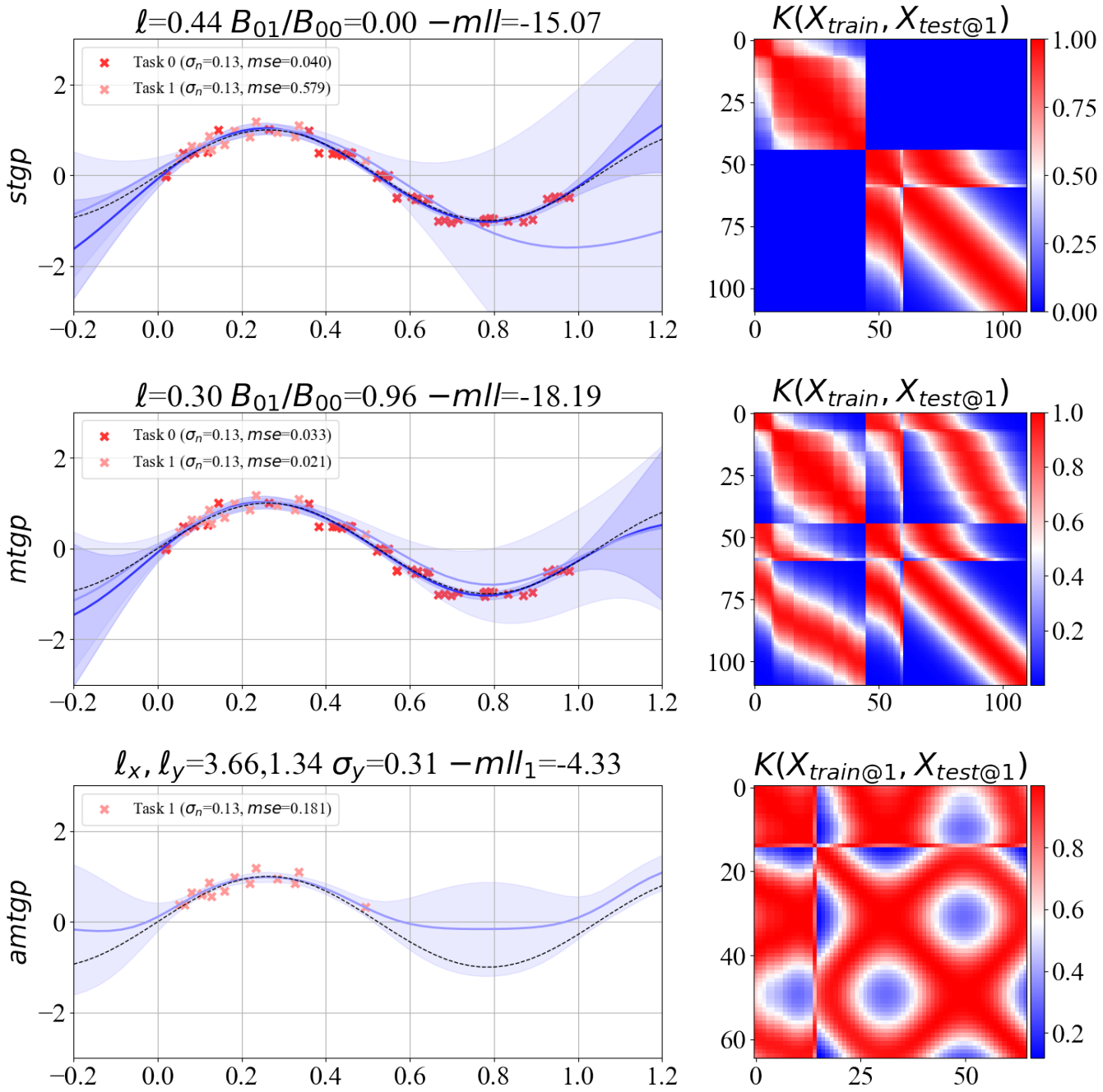

Figure 5: (Top) `stgp` GP regressors are learnt independently (Middle) `mtgp` GP regressors learnt via MTGP (Bottom) `amtgp` GP regressor for task 1 (main task) where the GP prior has kernel defined in Equation (14, 15). Missing side information is imputed from the predictive mean of a pre-trained `stgp` GP regressor for task 0. Bandwidth for the squared exponential kernel k_x and ARD kernel k_s, e.g. $\{\ell_x, \ell_s\}$, is optimized by maximizing the marginal likelihood of task 1 data points. We see, in case where the tasks are related, as measured by a large `shift` $= 0.$ betwen these two task functions. We see `mtgp` is able to utilize side information really effectively while `amtgp` also learns to rely on side information, as seen from a small ℓ_s, and reduce its uncertainty of prediction far away from its training support.

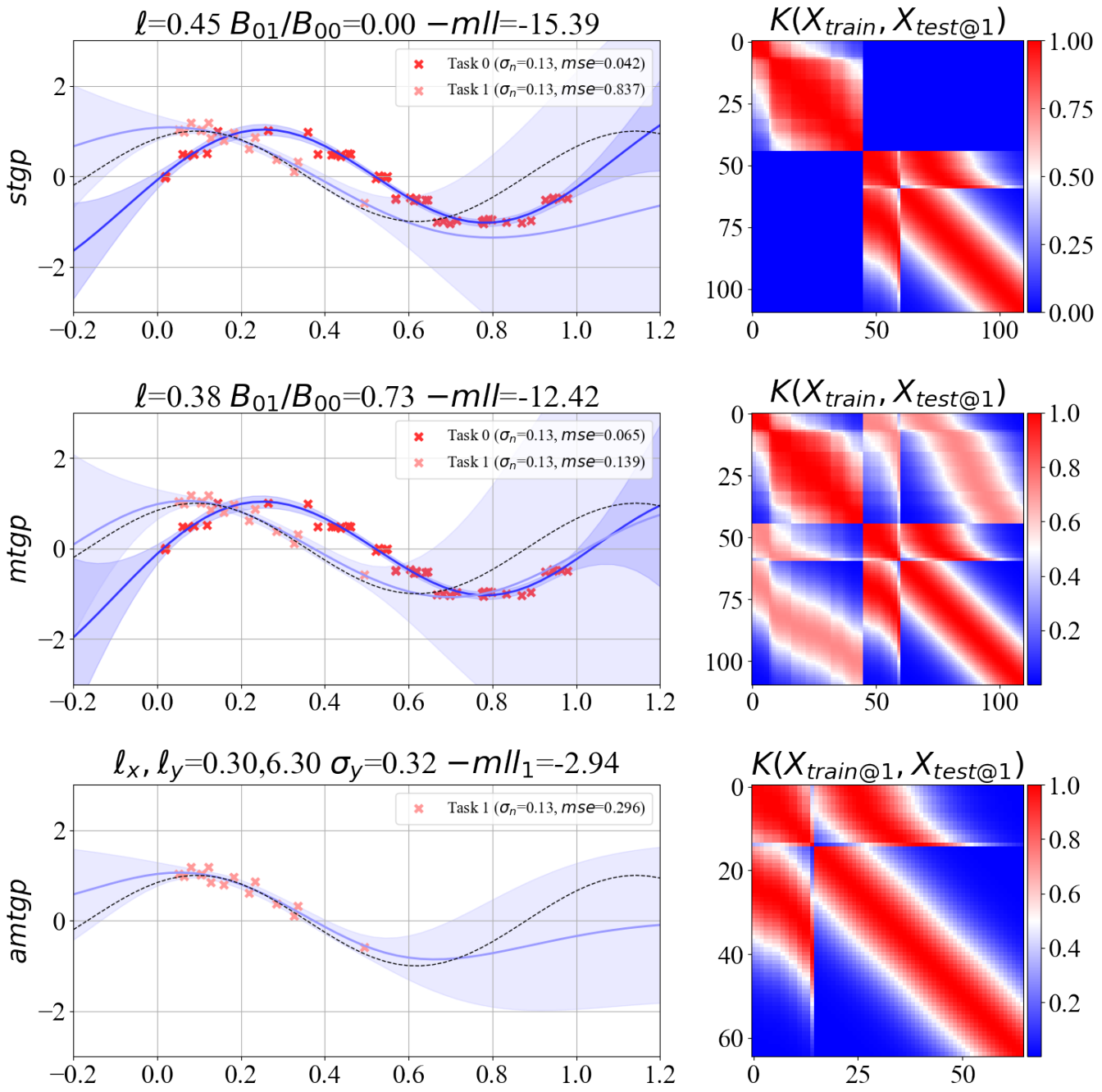

Figure 6: (Top) `stgp` GP regressors are learnt independently (Middle) `mtgp` GP regressors learnt via MTGP (Bottom) `amtgp` GP regressor for task 1 (main task) where the GP prior has kernel defined in Equation (14, 15). Missing side information is imputed from the predictive mean of a pre-trained `stgp` GP regressor for task 0. Bandwidth for the squared exponential kernel k_x and ARD kernel k_s, e.g. $\{\ell_x, \ell_s\}$, is optimized by maximizing the marginal likelihood of task 1 data points. We see, in case where the tasks are unrelated and perhaps even conflicting, as measured by a large `shift` = 1.0 between these two task functions, we see `mtgp` prediction is more uncertain in regions where task 1 function is supported and task 0 function is not. This is a result of `mtgp` able to use data points from task 0 observations even if the two functions are unrelated. `amtgp` learns to ignore side information as seen from large values of ℓ_s, and reduces to `stgp` regression

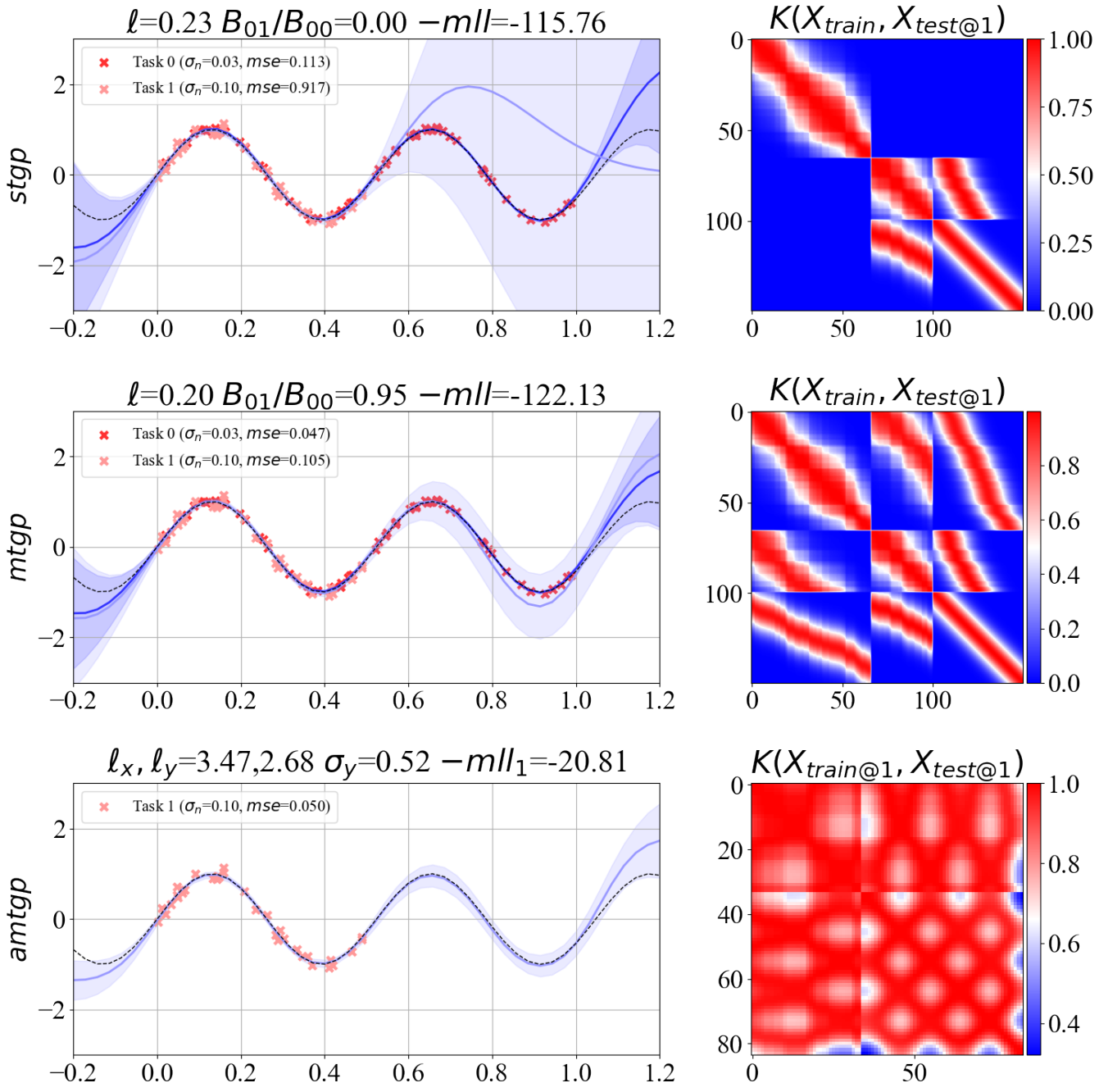

Figure 7: (Top) `stgp` GP regressors are learnt independently (Middle) `mtgp` GP regressors learnt via MTGP (Bottom) `amtgp` GP regressor for task 1 (main task) where the GP prior has kernel defined in Equation (14, 15). We see `amtgp` can extrapolate training support by utilizing prediction of side task.