Noname manuscript No.
(will be inserted by the editor)

Focused multi-task learning in a Gaussian process framework

Gayle Leen* · **Jaakko Peltonen*** · **Samuel Kaski**

Received: date / Accepted: date

Abstract Multi-task learning, learning of a set of tasks together, can improve performance in the individual learning tasks. Gaussian process models have been applied to learning a set of tasks on different data sets, by constructing joint priors for functions underlying the tasks. In these previous Gaussian process models, the setting has been symmetric in the sense that all the tasks have been assumed to be equally important, whereas in settings such as transfer learning the goal is *asymmetric*, to enhance performance in a target task given the other tasks. We propose a focused Gaussian process model which introduces an "explaining away" model for each of the additional tasks to model their non-related variation, in order to focus the transfer to the task-of-interest. This focusing helps reduce the key problem of *negative transfer*, which may cause performance to even decrease if the tasks are not related closely enough. In experiments, our model improves

G. Leen
Helsinki Institute for Information Technology HIIT
Department of Information and Computer Science
Aalto University
Currently at: deCODE Genetics, Reykjavik
E-mail: gayle.leen@decode.is

J. Peltonen
Helsinki Institute for Information Technology HIIT
Department of Information and Computer Science
Aalto University
P.O. Box 15400, FI-00076 Aalto, Finland
E-mail: jaakko.peltonen@aalto.fi

S. Kaski
Helsinki Institute for Information Technology HIIT
Department of Information and Computer Science
Aalto University
P.O. Box 15400, FI-00076 Aalto, Finland
and at: Helsinki Institute for Information Technology HIIT
Department of Computer Science
University of Helsinki
P.O. Box 15400, FI-00014 University of Helsinki, Finland
E-mail: samuel.kaski@hiit.fi

* Equal contribution.

performance compared to single-task learning, symmetric multi-task learning using hierarchical Dirichlet processes, transfer learning based on predictive structure learning, and symmetric multi-task learning with Gaussian processes.

Keywords Gaussian processes, multi-task learning, transfer learning, negative transfer

1 Introduction

In classification and regression tasks it is common that there are too few training data available from the task of interest to learn a good model for the task. Trying to learn a flexible model with many parameters from few data may result in overlearning where the model mistakes artifacts of the specific available samples as actual properties of the underlying distribution; alternatively, a simple model with few parameters might yield less overlearning but may also be unable to represent the properties of the distribution needed for good classification or regression performance. Learning the classification or regression model from the data of the current task alone is called *single-task learning*.

The problem of having too few data is particularly pressing in data-analysis settings characterized by the "small n, large p" problem of having a large dimensionality p and small sample size n. In this paper we will use functional neuroimaging as one of the case studies, and in functional Magnetic Resonance Imaging (fMRI) in particular the number of volume elements (voxels) p in which brain activity is measured is huge. Another well-known example of "small n, large p" conditions is genome-wide measurements of gene expression or other cellular data, where it may be of interest to measure a large number of variables p (for instance genes) in parallel. In these applications the number of samples n (the number of stimuli per subject in an fMRI study or the number of biological samples in a gene expression study) is small because of the cost of one measurement, or availability of relevant subjects or samples. In patient studies of a brain disorder, for instance, there are practical limitations on how many patients can be accessed and measured, and in experimental neuroscience the problem is that the larger the number of replications and variants needed, the less new neuroscience can be done per measurement.

Gaining more data from related tasks. The few training data available from the task of interest are *representative* in the sense that they are typically assumed to come from the same distribution as future test data. Even though there are few representative data available from the task of interest, there may be more data available from other *potentially related tasks*. The distribution of data in these other tasks is not the same as in the task of interest, but may be similar to it. If the distributions in several tasks are similar to each other, it may be possible to use the data from the other tasks to help learn each individual task. It has been shown that transferring knowledge between several potentially related learning tasks can improve performance. This scenario, termed *multi-task learning* (Caruana, 1997) or *transfer learning* (Thrun, 1996), has gained considerable attention in the machine learning community in recent years (see Pan and Yang (2010) for a recent review).

Transfer has usually been studied for regression or classification tasks; multi-task regression and classification scenarios arise in several application domains. For

example, if the task of interest is to classify gene expression profiles of patients as having a particular cancer type, then other data sets of patients having other cancer types can be related tasks. If the task is to classify scientific articles from a conference into subject categories, then classifying articles from related conferences can by used as related tasks. In this paper we study a classification task in which the goal is to predict the stimulus given brain measurements of a certain user, utilizing the measurements of other users on the same and different stimuli as related tasks; in this setting a multi-task learning setup is useful because, when generalizing across subjects, the brain physiology and function are sufficiently similar that different brains can be matched, but the matching is only approximate.

Multi-task learning by hierarchical modeling. Learning from several tasks is often done by constructing a *hierarchical model* over all tasks, where model parameters within each task are related to the corresponding parameters in the other tasks through an upper-level prior distribution. For example, if there are several related linear regression tasks, they can be learned together by assuming their regression weights are drawn from a common upper-level prior, which effectively constrains the weights to be similar across the tasks. Data from all tasks then affect the learning of the upper-level prior parameters, which in turn affect the learning of the parameters within each task; this effectively provides additional indirect evidence for learning the parameters of each task. Parameters in each task may then be learned closer to their actual underlying values, or in Bayesian learning may be inferred with less remaining uncertainty. This kind of learning process across tasks is sometimes called *sharing statistical strength* (Teh et al, 2005, 2006). Sharing statistical strength between tasks can potentially compensate for having very few samples in the desired learning task, and can make the inference more robust to noise.

Negative transfer. Learning several tasks together may not always be beneficial. As usual, both single-task learning and multi-task learning can suffer from misspecifying the model within the tasks, however, in multi-task settings there is an additional potential danger: in order to enable learning from data across the tasks, assumptions must be made about the possible model relationships between tasks. Transfer of knowledge between different tasks is useful only when the tasks are related; misspecifying the possible kinds relationships between model parameters across tasks, or misspecifying relationships to be likely where they are unlikely in reality, can distort the the model learned for a target task rather than providing additional statistical strength. The phenomenon where providing other tasks to help learning actually ends up hurting the learning is called *negative transfer* (see, e.g., Rosenstein et al 2005).

Negative transfer can happen in particular if the distributions in some of the other tasks are in reality not similar to a task of interest: then learning the tasks together in a hierarchical fashion, and assuming the parameters to be similar, can harmfully constrain the parameters for the task of interest. The precise effect of the negative transfer depends on the assumed task relationships and the model families used within the tasks. The harmful effect will typically be strongest for inputs where observations from the other tasks strongly outnumber those from the

task of interest. For such inputs, the model learned for the task of interest may mistakenly predict outputs to be similar to the other tasks.

A crucial part of multi-task learning algorithms lies in the modelling of task relatedness, through the specification and the learning of the dependency structure between tasks. We discuss learning the dependency structure in an asymmetric setting: we propose a method that aims to avoid negative transfer by a flexible dependency structure where the dependency can be different from a task of interest to each other task, and even to different parts of the other tasks.

1.1 Symmetric and Asymmetric Multi-task Learning

In general, existing multi-task learning approaches use a symmetric dependency structure between tasks. This type of set-up, which we term *symmetric multi-task learning*, assumes that all tasks are of equal importance. The set of related tasks is learned jointly, with the aim of improving over learning the tasks separately (the *no transfer* case), averaged over all tasks.

However, a common learning scenario is to learn a specific task (*primary task*), while incorporating knowledge learned through other similar tasks (*secondary tasks*). An asymmetric scenario is natural especially when future test data will come only from the task of interest; for example, the term *transfer learning* is often used to denote setting where several tasks have been learned at an earlier time and their knowledge is transferred to help a new task at hand. In one transfer learning setting, the task of interest may be to classify scientific papers for the current and next year of a particular conference, whereas the data of the related other tasks may be documents from earlier years of the conference when the topics prevalent in the papers were different, and from other conferences where even the scopes of the conferences are different. When classifying presence of a particular disease in patients based on gene expression, historical data sets from related other diseases may be used as related tasks but the aim is to learn to classify the disease of interest rather than the other diseases which are only used as sources of related information. In the neuroscience scenario that we use as a case study in this paper, we are interested in learning about a specific patient's response to a stimulus, but we can transfer information from other patients' responses to related stimuli to improve learning. The data of the other patients may be historical measurements from persons who are not currently participating in the neuroscience study. An asymmetric setting can also happen in multi-task learning when tasks are learned simultaneously, but one of them is more interesting than others: for example, in a neuroscience scenario one task may be to detect an interesting stimulus based on the brain response whereas other tasks may be to detect ordinary stimuli.

The asymmetric learning setting requires the assumption of an asymmetric dependency structure between tasks. Existing approaches include reweighting-based methods (Wu and Dietterich, 2004; Bickel et al, 2008, 2009) or learning of shared feature spaces. An alternative has been to, in effect, use a symmetric multi-task learning method in an asymmetric mode, by using the model learned from auxiliary tasks as a prior for the target task (Marx et al, 2005; Raina et al, 2005; Xue et al, 2007).

Inspired by the Gaussian process (GP) models used earlier for symmetric multi-task learning, we propose a novel and simple dependency structure for asymmetric

multi-task learning using GPs. This focuses on learning a target task and learns to avoid negative transfer; this can be done conveniently in the GP formulation, by adding task-specific processes which "explain away" irrelevant properties. At the same time, flexibility of the GP framework is preserved.

2 Dependency Structure in Multi-task Learning with Gaussian Processes

Supervised learning tasks such as classification and regression can be viewed as function approximation problems given the task inputs and targets; accordingly, multi-task learning can be viewed as learning multiple related functions. The Gaussian process (GP) framework provides a principled and flexible approach for constructing priors over functions.

In brief, a GP is a prior over input-output functions that does not restrict outputs to a particular parametric function of input coordinates (such as a sinusoid or a polynomial); instead, for any fixed set of input points the prior for the corresponding set of outputs is represented as a multidimensional Gaussian distribution. The GP prior over the whole input-output function is then specified by a mean function and a covariance function; often a zero-mean prior is used. Typical covariance functions such as the squared exponential covariance function specify that nearby inputs should *a priori* have strongly related outputs. Given the GP prior and a set of observed input-output samples, Bayesian inference is used to infer the posterior over the possible underlying functions. If the observation model is Gaussian the inference of the posterior can be done analytically. The inferred posterior can be used for example to predict output values. Even though the mean and covariance function that specify the GP prior can be fairly simple functions, the inferred posterior can very flexibly represent complicated functions.

The GP framework has subsequently been applied successfully to multi-task learning problems (Yu and Tresp, 2005; Bonilla et al, 2008; Alvarez and Lawrence, 2009). A crucial element of these models is the way in which the dependency structure between the multiple functions is encoded through the construction of the covariance function. However, current GP approaches do not address the problem of asymmetric multi-task learning, and only consider symmetric dependency structures, which we review in the following subsection.

2.1 Symmetric Dependency Structure

Suppose that there are N distinct inputs, $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_N]^\top$, and M tasks, such that y_i^t is the target for input i in task t. We denote the vector of outputs for task t as $\mathbf{y}^t = [y_1^t, ..., y_N^t]^\top$, and the $N \times M$ vector of outputs for all M tasks, as $\mathbf{y} = [(\mathbf{y}^1)^\top, ..., (\mathbf{y}^M)^\top]^\top$. Here we consider a set of tasks which all have the same input, for ease of notation, but the problem setting can easily be generalised to different inputs for each task. In the GP approach to the problem, it is assumed that there is a latent function underlying each task, $f^1, ..., f^M$. Denoting the latent function evaluated at input i for task t as $f^t(\mathbf{x}_i)$, a (zero mean) GP prior is defined over the latent functions, with a covariance function of the form

$$\left\langle f^t(\mathbf{x}) f^{t'}(\mathbf{x}') \right\rangle = k^T(t, t') k^x(\mathbf{x}, \mathbf{x}') \tag{1}$$

where $\langle\cdot\rangle$ denotes expectation and $\langle f^t(\mathbf{x})f^{t'}(\mathbf{x}')\rangle$ is the usual definition of a covariance function, expectation of the product of the two outputs $f^t(\mathbf{x})$ and $f^{t'}(\mathbf{x}')$, where the expectation is taken over the Gaussian process prior. On the right-hand side k^T is a covariance function over tasks, specifying the intertask similarities, and k^x is a covariance function over inputs. For regression tasks, the observation model is $y_i^t \sim \mathcal{N}(f^t(\mathbf{x}_i), \sigma_t^2)$, where σ_t^2 is the noise variance in task t.

The covariance function k^x over inputs can be any typical function used in Gaussian processes, such as a radial basis function $k^x(\mathbf{x}, \mathbf{x}') \propto \exp(-||\mathbf{x}-\mathbf{x}'||_U^2/2)$ where U denotes a metric, usually a Euclidean or Mahalanobis metric. In the experiments we use such a radial basis function, detailed in Section 5.

Bonilla et al (2008) define k^T as a "free-form" covariance function, where $k^T(i,j) = \mathbf{K}^T_{i,j}$ indexes a positive semidefinite intertask similarity matrix $\mathbf{K}^T$. Other methods such as that of Yu et al (2007) have included a parameterised similarity matrix over task descriptor features, but this could be restrictive in modelling similarities between tasks. These types of priors essentially assume that each of the task latent functions is a linear combination of a further set of latent functions, known as intrinsic correlation models in the geostatistics field (see, e.g., Wackernagel 1994). This idea was further generalised by Alvarez and Lawrence (2009) to generating the task latent functions by convolving a further set of latent functions with smoothing kernel functions.

2.2 Predictive Mean for Symmetric Multi-task GP

The predictive mean on a new data point $\mathbf{x}_*$ in task j, for the multi-task GP formulation of Bonilla et al (2008), is given by

$$\overline{f}^j(\mathbf{x}_*) = (\mathbf{k}_j^T \otimes \mathbf{k}_*^x)^\top \boldsymbol{\Sigma}^{-1}\mathbf{y} \quad \text{where} \quad \boldsymbol{\Sigma} = \mathbf{K}^T \otimes k^x(\mathbf{X},\mathbf{X}) + \mathbf{D} \otimes \mathbf{I} \tag{2}$$

where $\mathbf{k}_j^T$ is the jth column of task similarity matrix $\mathbf{K}^T$, $\otimes$ is the Kronecker product, and $\mathbf{k}_*^x = [k(\mathbf{x}_*, \mathbf{x}_1), ..., k(\mathbf{x}_*, \mathbf{x}_N)]^\top$ is the vector of covariances between the test input $\mathbf{x}_*$ and the training inputs. The $k^x(\mathbf{X},\mathbf{X})$ is the matrix of covariance function values between all training input points, and $\mathbf{D}$ is an $M \times M$ diagonal matrix where the (t,t)th element is σ_t^2.

To gain intuition into the form of the predictive mean, let us define the $M \times N$ vector $\mathbf{w} = \boldsymbol{\Sigma}^{-1}\mathbf{y}$, and divide it into M blocks of N elements: $\mathbf{w} = [\mathbf{w}_1^\top, ..., \mathbf{w}_M^\top]^\top$. We can then rewrite (2) as

$$\overline{f}^j(\mathbf{x}_*) = \sum_{m=1}^{M} \mathbf{K}^T_{m,j}(\mathbf{k}_*^x)^\top \mathbf{w}_m = \sum_{m=1}^{M} \mathbf{K}^T_{m,j}\mu_*^m \tag{3}$$

where $\mu_*^m = (\mathbf{k}_*^x)^\top \mathbf{w}_m$ can be interpreted as the posterior mean of the latent function at $\mathbf{x}_*$ for task m; thus (2) is a weighted sum of posterior means for all tasks, and the weights $\{\mathbf{K}^T_{m,i}\}_{m=1}^M$ are covariances between task j and all tasks. Since $\mathbf{K}^T$ is positive semidefinite, the sharing of information between tasks is naturally symmetric, and all tasks are treated equally. However, we are interested in an asymmetric setup, where we learn a primary task together with several secondary tasks. Rather than modelling the relationships between secondary tasks, we want to focus on the aspects relevant to learning the primary task.

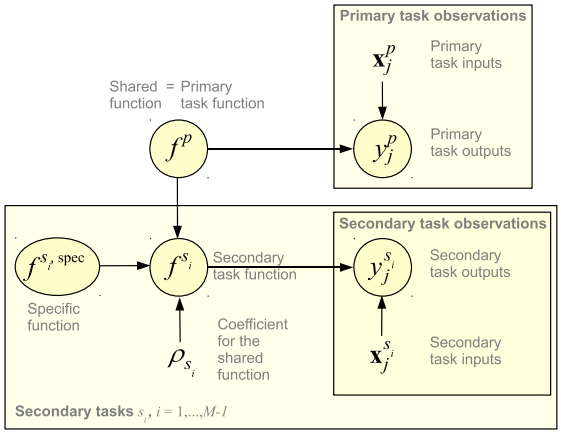


Fig. 1 Graphical model of the focused GP multi-task model, showing the relationship between the function values of the primary and secondary tasks. Parameters of the covariance functions omitted for clarity.

2.3 Asymmetric Dependency Structure

In the previous symmetric learning problem, the tasks were modelled as conditionally independent on a set of M (i.i.d.) underlying functions, which capture the shared structure between all tasks. In this section, we derive an asymmetric version of a GP framework for multi-task learning, by constraining the secondary tasks to be conditionally independent given the primary task, such that the shared structure between all secondary tasks is due to the primary task function.

Similarly to the previous notation, let us denote the inputs to each task as $\mathbf{X}$. Suppose that there is one primary task, with targets $\mathbf{y}^p = [y_1^p, ..., y_N^p]^\top$, with underlying latent function values $\mathbf{f}^p = [f^p(\mathbf{x}_1), ..., f^p(\mathbf{x}_N)]^\top$. Suppose there are $M-1$ secondary tasks, where the targets for the ith secondary task are denoted by $\mathbf{y}^{s_i} = [y_1^{s_i}, ..., y_N^{s_i}]^\top$. The corresponding latent function values are $\mathbf{f}^{s_i} = [f^{s_i}(\mathbf{x}_1), ..., f^{s_i}(\mathbf{x}_N)]^\top$.

We are interested in learning the underlying function f^p for the primary task. Here, potentially related secondary tasks can help to learn f^p; conversely if we know f^p, this could help to learn the functions underlying the secondary tasks $\{f^{s_i}\}$. First we define a joint prior over the primary and secondary task function values. We start by making the assumption that the secondary task functions $\{f^{s_i}\}$ can be decomposed into a "shared" component (which is shared with the primary task) and a "specific" component. That is, for the nth input,

$$f^{s_i}(\mathbf{x}_n) = f^{s_i,\text{shared}}(\mathbf{x}_n) + f^{s_i,\text{specific}}(\mathbf{x}_n) \; . \tag{4}$$

Further we assume that $f^{s_i,\text{shared}} = \rho_{s_i} f^p$, that is, the shared component is correlated with the primary task function. This may seem like a restrictive assumption but assuming linear relationships between task functions has been proved to be successful by, e.g., Wackernagel (1994) and Bonilla et al (2008). Now we can place a shared prior over each $f^{s_i,\text{shared}}$ and f^p. The corresponding graphical model is presented in Figure 1.

2.3.1 Sharing between Primary and Secondary Task Functions.

Since the functions in the secondary tasks are composed of a shared and specific component as shown in (4), we can define the covariance function separately for both types of components. We first discuss the covariance between the primary function f^p and the shared components $f^{s_i,\text{shared}}$ of the secondary tasks s_i.

Let t and t' be indices of two tasks, each of which can be the primary task or one of the secondary tasks. We place a zero mean Gaussian process prior on f^p, with covariance function k^p, such that the prior on the shared function is also a GP, with covariance function

$$\left\langle f^t(\mathbf{x}) f^{t'}(\mathbf{x}') \right\rangle = k^T(t,t') k^p(\mathbf{x},\mathbf{x}') \quad \text{where} \quad k^T(t,t') = \rho_t \rho_{t'} \tag{5}$$

where ρ_t is the correlation of task t with the primary task, and $\rho_p = 1$, and f^t can denote either the primary task function or the shared component in any of the secondary tasks. Denoting the shared components of the task functions for the $M-1$ secondary tasks as $\mathbf{f}^{s,\text{shared}} = \left[(\mathbf{f}^{s_1,\text{shared}})^\top, ..., (\mathbf{f}^{s_{M-1},\text{shared}})^\top \right]^\top$, the joint distribution over the shared function values is given by

$$p(\mathbf{f}^p, \mathbf{f}^{s,\text{shared}}) = \mathcal{GP}\left(\begin{bmatrix} \mathbf{f}^p \\ \mathbf{f}^{s,\text{shared}} \end{bmatrix}; 0, \begin{bmatrix} \mathbf{K}_{pp} & \mathbf{K}_{sp}^\top \\ \mathbf{K}_{sp} & \mathbf{K}_{ss} \end{bmatrix} \right) \tag{6}$$

where the expression on the right-hand side is of the form $\mathcal{GP}(\mathbf{f}; 0, \mathbf{K})$ which denotes a Gaussian process prior with mean 0 and covariance matrix $\mathbf{K}$ evaluated at function value $\mathbf{f}$. Here in particular $\mathbf{K}_{pp}$ is the matrix of covariance function values from (5) between the primary task points, $\mathbf{K}_{sp}$ evaluated between secondary and primary, and $\mathbf{K}_{ss}$ between secondary task inputs, where the matrices $\mathbf{K}_{sp}$ and $\mathbf{K}_{ss}$ represent variation due to the shared components in the secondary tasks.

2.3.2 Explaining Away Secondary Task-Specific Variation.

We next treat the covariance between the specific components of the secondary tasks, and then put the types of covariances together to form the total covariance between the tasks.

We define the covariance function over $\mathbf{f}^{s,\text{specific}} = \left[(\mathbf{f}^{s_1,\text{specific}})^\top, ..., (\mathbf{f}^{s_{M-1},\text{specific}})^\top \right]^\top$ to be block diagonal in $[\mathbf{K}_1^{\text{spec}}, ..., \mathbf{K}_{M-1}^{\text{spec}}]$ with respect to the tasks; we denote the resulting block diagonal covariance matrix over the secondary tasks as $\mathbf{K}^{\text{spec}}$. The covariance functions $\mathbf{K}_{s_i}^{\text{spec}}$ have parameters specific to each secondary task s_i, and the specific functions over all secondary tasks are then drawn as $\mathbf{f}^{s,\text{specific}} \sim \mathcal{GP}(0, \mathbf{K}^{\text{spec}})$. This creates flexible models for the secondary tasks, which can "explain away" variation that is specific to a secondary task, and unshared with the primary task. The full secondary task functions are then generated according to equation (4) as $\mathbf{f}^s = \mathbf{f}^{s,\text{shared}} + \mathbf{f}^{s,\text{specific}}$; since the shared components are independent of the specific components, the covariance of the full functions is just the sum of covariances of the shared and specific components. The model, which we call the Focused GP-multitask learning model, takes the form

$$p(\mathbf{f}^p, \mathbf{f}^s) = \mathcal{GP}\left(\begin{bmatrix} \mathbf{f}^p \\ \mathbf{f}^s \end{bmatrix}; 0, \begin{bmatrix} \mathbf{K}_{pp} & \mathbf{K}_{sp}^\top \\ \mathbf{K}_{sp} & \mathbf{K}_{ss} + \mathbf{K}^{\text{spec}} \end{bmatrix} \right) . \tag{7}$$

2.4 Sparse approximation to the focused GP-multitask learning model

Learning the hyperparameters for the covariance functions in (7) will be computationally expensive since it involves the inversion of the full covariance matrix of the Gaussian process prior across all points in all tasks, which is a matrix of size $(M \times N) \times (M \times N)$. Inverting such a matrix takes $O(M^3N^3)$ time in the general case; however, if the matrix has a sufficiently simple structure the inversion can be computed faster. In this section, we derive an approximation to this covariance matrix based on assumptions about the sharing between secondary and primary tasks. The idea is to approximate the matrix, preserving the main part of our intended asymmetric multitask dependency structure, but simplifying it enough so that we can apply the *Woodbury identity* to compute the inversion. Note that this approximation is not crucial to our method: if there are few enough data to compute the full inverse, the approximation can be omitted.

To start, note that the covariance matrix across all the tasks has a block matrix form, and a block matrix can be inverted as

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}^{-1} = \begin{bmatrix} (\mathbf{A}-\mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1} & -(\mathbf{A}-\mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1}\mathbf{B}\mathbf{D}^{-1} \\ -\mathbf{D}^{-1}\mathbf{C}(\mathbf{A}-\mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1} & \mathbf{D}^{-1}\mathbf{C}(\mathbf{A}-\mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1}\mathbf{B}\mathbf{D}^{-1}+\mathbf{D}^{-1} \end{bmatrix}$$

where $\mathbf{A}$ corresponds to the covariance block within the primary task, $\mathbf{D}$ corresponds to the covariance block between the secondary tasks, and $\mathbf{B}$ and $\mathbf{C}$ are the cross-covariance blocks from the primary task to the secondary tasks. On the right-hand side, the only large matrix that needs to be inverted is $\mathbf{D}$ which corresponds to the covariance between secondary tasks; we must find an approximation for this covariance that will be efficient to invert.

To find the approximation, first note that the value of a GP prior over the primary and secondary functions $\mathbf{f}^p$ and $\mathbf{f}^s$ can be evaluated as the product of the value of a conditional prior and the value of a marginal prior, so that $p(\mathbf{f}^p, \mathbf{f}^s) = p(\mathbf{f}^s|\mathbf{f}^p)p(\mathbf{f}^p)$. In particular, if the GP prior is of the form in (7) then by standard Gaussian identities we can write it in the equivalent form

$$p(\mathbf{f}^p, \mathbf{f}^s) = p(\mathbf{f}^s|\mathbf{f}^p)p(\mathbf{f}^p) = \mathcal{GP}\left(\mathbf{f}^s; \mathbf{K}_{sp}\mathbf{K}_{pp}^{-1}\mathbf{f}^p, \boldsymbol{\Lambda} + \mathbf{K}^{\text{spec}}\right)\mathcal{GP}(\mathbf{f}^p; 0, \mathbf{K}_{pp}) \quad (8)$$

where $\boldsymbol{\Lambda} = \mathbf{K}_{ss} - \mathbf{K}_{sp}\mathbf{K}_{pp}^{-1}\mathbf{K}_{sp}^{\top}$. The first GP term on the right-hand side is

$$p(\mathbf{f}^s|\mathbf{f}^p) = \mathcal{GP}\left(\mathbf{f}^s; \mathbf{K}_{sp}\mathbf{K}_{pp}^{-1}\mathbf{f}^p, \boldsymbol{\Lambda} + \mathbf{K}^{\text{spec}}\right) , \quad (9)$$

which is the GP predictive likelihood on the secondary task function values, after training on the primary task. The second GP term on the right-hand side is simply the marginal prior $p(\mathbf{f}^p)$ in the primary task. We will now approximate the first term by a simpler form, by approximating $\boldsymbol{\Lambda}$ as a diagonal matrix: we simply set the diagonal of $\boldsymbol{\Lambda}$ to the diagonal of $\mathbf{K}_{ss} - \mathbf{K}_{sp}\mathbf{K}_{pp}^{-1}\mathbf{K}_{sp}^{\top}$, and set off-diagonal elements of $\boldsymbol{\Lambda}$ to zero. Then the total covariance matrix $\boldsymbol{\Lambda} + \mathbf{K}^{\text{spec}}$ in the conditional prior $p(\mathbf{f}^s|\mathbf{f}^p)$ is a simple block-diagonal matrix.

In order to use the approximation in GP learning, we must recover the block-matrix representation of the full GP prior over $\mathbf{f}^p$ and $\mathbf{f}^s$, similar to equation (7) but with the approximation taken into account. To do this we must simply recompute the marginal covariance matrix of $\mathbf{f}^s$, as the integral $p(\mathbf{f}^s) = \int_{\mathbf{f}^p} p(\mathbf{f}^s|\mathbf{f}^p)p(\mathbf{f}^p)d\mathbf{f}^p$. This yields

$$p(\mathbf{f}^s) = \mathcal{GP}\left(0, \mathbf{K}_{sp}\mathbf{K}_{pp}^{-1}\mathbf{K}_{sp}^{\top} + \boldsymbol{\Lambda} + \mathbf{K}^{\text{spec}}\right) . \quad (10)$$

This yields the final prior on all the task functions as

$$p(\mathbf{f}^p, \mathbf{f}^s) = \mathcal{GP}\left(0, \begin{bmatrix} \mathbf{K}_{pp} & \mathbf{K}_{sp}^\top \\ \mathbf{K}_{sp} & \mathbf{K}_{sp}\mathbf{K}_{pp}^{-1}\mathbf{K}_{sp}^\top + \boldsymbol{\Lambda} + \mathbf{K}^{\text{spec}} \end{bmatrix}\right). \tag{11}$$

Since the above prior uses the reduced rank (the rank = number of primary task inputs) approximation to the covariance matrix, we can use the Woodbury identity to efficiently calculate the inverse and determinant. In particular, by the Woodbury identity the inverse of the bottom-right block is

$$(\boldsymbol{\Lambda}+\mathbf{K}^{\text{spec}})^{-1} - (\boldsymbol{\Lambda}+\mathbf{K}^{\text{spec}})^{-1}\mathbf{K}_{sp}[\mathbf{K}_{pp}+\mathbf{K}_{sp}^\top(\boldsymbol{\Lambda}+\mathbf{K}^{\text{spec}})^{-1}\mathbf{K}_{sp}]^{-1}\mathbf{K}_{sp}^\top(\boldsymbol{\Lambda}+\mathbf{K}^{\text{spec}})^{-1}$$

which is efficient to compute since $\boldsymbol{\Lambda}+\mathbf{K}^{\text{spec}}$ is block-diagonal and the term inside the brackets is a small matrix of the same size as $\mathbf{K}_{pp}$, having the same number of rows as there are samples in the primary task. This inverse can then be inserted in the general block-matrix inverse equation, to yield the inverse of the complete covariance matrix. A similar efficient computation can be done to compute the determinant. We call this a "sparse" approximation because several entries of $\boldsymbol{\Lambda}$ were approximated as zero to reduce the rank of the full covariance matrix; however, note that the reduced-rank full matrix itself in (11) remains non-sparse.

2.4.1 Influence of the primary observations on secondary task predictions

In addition to deriving approximations, the conditional prior (predictive likelihood) equation in (9) is also useful for analyzing the behavior of the asymmetric learning. An interpretation of equation (9) is that the secondary task functions are given by the posterior distribution of f^p (the primary task function) after observing the primary task function values $\mathbf{f}^p$, evaluated at all the secondary task inputs, with an added "specific" component modelled by $\mathbf{K}^{\text{spec}}$. The mean prediction $\mathbf{K}_{sp}\mathbf{K}_{pp}^{-1}\mathbf{f}^p$ is similar to a standard GP predictive equation, with the difference that according to the definition of $\mathbf{K}_{sp}$ in (5) the posterior mean for each secondary task s is weighted by ρ_s, which models the correlation with the primary task. To illustrate this, for secondary task l, the posterior mean $\bar{\mathbf{f}}^{l,\text{shared}}$ given $\mathbf{f}^p$ becomes

$$\bar{\mathbf{f}}^{l,\text{shared}} = \rho_l k^p(\mathbf{X}_l, \mathbf{X}_p)k^p(\mathbf{X}_p, \mathbf{X}_p)^{-1}\mathbf{f}^p = \rho_l \boldsymbol{\mu}_l^p \tag{12}$$

where we have used the notation: $\mathbf{X}_i$ is the set of input points for task i, and $\boldsymbol{\mu}_l^p$ is the posterior mean given covariance function k^p and observations $\mathbf{f}^p$, evaluated at $\mathbf{X}_l$. Controlling ρ_s therefore directly controls the amount of influence the primary task predictions have on predictions in the secondary tasks, and hence the amount of influence the secondary task observations have on learning the primary task function. Learning ρ_s during training can help to avoid negative transfer from secondary tasks to the primary task.

2.5 Hyperparameter Learning

We can learn the hyperparameters of our model in (11) by optimising the marginal log likelihood with respect to the hyperparameters of the covariance functions (the hyperparameters of the covariance function of the shared components and the corresponding hyperparameters of the covariance function of the specific components),

the task similarity vector $[\rho_{s_1}, ..., \rho_{s_{M-1}}]$, and the parameters of the observation model, given the inputs $\mathbf{x}$ and targets y. The hyperparameters needed for the covariance functions depend on the form of the covariance function; in the experiments we use a squared exponential form described in Section 5. For regression, the observation model is $y_i^t \sim \mathcal{N}(f^t(\mathbf{x}_i), \sigma_t^2)$, where σ_t^2 is the noise variance in task t. The log marginal likelihood of the observed data has the same form as usual in Gaussian process regression: denoting the vector of observed outputs over all tasks as $\mathbf{y}$ and the corresponding matrix of inputs as $\mathbf{X}$, we have

$$L = \log p(\mathbf{y}|\mathbf{X}) = -\frac{1}{2}\log|\boldsymbol{\Sigma}| - \frac{1}{2}(\mathbf{y}-\mathbf{0})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{y}-\mathbf{0}) - \frac{N}{2}\log(2\pi) \tag{13}$$

where $\boldsymbol{\Sigma} = \mathbf{K} + \boldsymbol{\Sigma}_{\text{noise}}$, $\mathbf{K}$ is the covariance matrix on the right-hand side of (11) and $\boldsymbol{\Sigma}_{\text{noise}}$ is a diagonal matrix where the diagonal entries corresponding to task t are σ_t^2. If all tasks have the same number of inputs then $\boldsymbol{\Sigma}_{\text{noise}}$ is the same as $\mathbf{D}\otimes\mathbf{I}$ in equation (2). The hyperparameters can be learned by optimising the above marginal log-likelihood with respect to the hyperparameters, which can be done by gradient methods (here we used standard conjugate gradient optimization).

After the parameters have been learned, the predictive mean for a new data point $\mathbf{x}_*$ from the primary task is given by:

$$f^p(\mathbf{x}_*) = k^{\text{shared}}(\mathbf{x}_*, \mathbf{X})\boldsymbol{\Sigma}^{-1}\mathbf{y} \tag{14}$$

where $k^{\text{shared}}(\mathbf{x}_*, \mathbf{X})$ is the vector of covariances between the test input and the shared functions in the training inputs (primary and all secondary inputs): for training input $\mathbf{x}$ in task t, the corresponding element in the vector has value $k^p(\mathbf{x}_*, \mathbf{x})k^T(p,t) = k^p(\mathbf{x}_*, \mathbf{x})\rho_t$.

The classification case is similar, we simply use a probit noise model $p(y_i^t \mid f_i^t) = \Phi(y_i^t(f_i^t + b))$, where f_i^t is the predicted function value for point i in task t, Φ is the cumulative distribution function for a standard Gaussian $\mathcal{N}(0,1)$, and b is a bias parameter. For the binary classification experiments in Section 5.2, we make an approximation to the model likelihood using Expectation Propagation (Minka, 2001).

3 Related Work and Discussion

The focused multi-task GP model that we have derived in the previous sections is designed for asymmetric multitask learning scenarios; we construct a joint GP prior over the functions underlying the tasks, that assumes an asymmetric dependency structure. Our approach uses a simple idea to bias the model to focus on learning the underlying function for the primary task, rather than modeling and learning all the tasks symmetrically. The dependency structure does this by decomposing the underlying task functions for the secondary tasks as "shared" and "specific" components. The shared components are from a joint GP prior with the primary task function. These are conditioned on the primary task function values according to equation (9) which biases the shared variation between tasks to be due to the primary task function, and a task specific weight which is learned during training. We additionally assume that each of the secondary task functions can

also be explained by a process specific to it, by defining a block diagonal covariance structure over the secondary tasks. This allows the model to "explain away" secondary task specific variation and focus the model on learning the primary task.

Recently there has been interest in asymmetrical GP multi-task learning (Chai, 2009), where generalisation errors for the multi-task GP of Bonilla et al (2008) were derived for an asymmetrical multi-task case, with one primary and one secondary task. However, this work did not derive a new model for asymmetric multi-task learning, and focused on analysing the symmetric model. In the next section we will analyse our asymmetric model in a similar manner.

When deriving the sparse approximation to our model, the sparse GP method of Snelson and Ghahramani (2006) bears similarities to our model. In Snelson and Ghahramani (2006), the covariance matrix of the GP was decomposed into a reduced rank matrix. This model assumes that there are a set of M pseudo-inputs, which along with their function values (pseudo-targets) act as a pseudo data set. This provides a compact summary of the real data (N data points; $M \ll N$). The covariance function is parameterised by the pseudo-input locations, which are learned during the optimization, by deriving the likelihood function for the real data as a predictive likelihood, given the pseudo data set. In our focused multi-task GP, we can interpret our sparse approximation as a special case of the sparse GP model; the pseudo-input locations are fixed as the inputs to the primary task, such that they are a compact representation of the shared function underlying the primary and secondary tasks. The distribution over the secondary task functions can be viewed as the predictive distribution given the primary task function values. In Section 2.4 we then assume the compact representation suffices to represent the shared component of variation inside the secondary tasks, so that the remaining off-diagonal elements in the matrix $\boldsymbol{\Lambda}$ were approximated as zero.

In this paper we make the simplifying assumption that the task of interest is entirely composed of the shared function, and that there are no other strong shared functions between other tasks. This model already proves useful in a challenging fMRI task, demonstrating that the idea of asymmetric modelling with explaining-away yields useful results, and it can be extended to more general asymmetric modelling in later stages. For instance, there may be detrimental shared variation between other tasks, which may harm learning of the primary task. In Section 6 we briefly study the effect of such detrimental shared variation on our current model. The model could be extended by adding additional GP functions which are shared between other tasks but not with the primary task. The overall model can then learn which shared function is a better explanation. As the number of tasks increases, the number of possible sharing configurations increases (shared functions between $2, 3, \ldots, M$ tasks) and the complexity of the model quickly increases. This will be studied in further work.

4 Examining the Generalisation Error for Asymmetric and Symmetric Models

To examine the effect of the processes that are specific to a secondary task, we look at the generalisation error on the primary task for the asymmetric two tasks case in a similar manner to Chai (2009). We investigate the influence of ρ, the degree of "relatedness" between the two tasks.

We want to study a continuum where single-task learning is one extreme, pooling all data into one task is the other extreme, and the asymmetric model lies in between them. Note that setting $\rho = 0$ in the asymmetric and symmetric cases reduces to single-task learning. For our model, we will study the case where the covariance of the specific function in the secondary tasks has its overall scale set to $(1 - \rho^2)$; then the extreme of $\rho = 1$ will reduce to pooling all data into one task, in both the symmetric and asymmetric cases. This corresponds to using an overall scale of 1 and multiplying the resulting specific covariance by $(1 - \rho^2)$; we use this notation to make the influence of ρ explicit.

Suppose that we have training inputs $\mathbf{X}_P$ for the primary task, and $\mathbf{X}_S$ for the secondary task. The covariance matrices $\mathbf{C}_{\text{sym}}$ and $\mathbf{C}_{\text{asym}}$, for the symmetric and asymmetric cases respectively, of the noisy training data are given by:

Symmetric case

$$\mathbf{C}_{\text{sym}}(\rho) = \mathbf{K}^{\text{sym}}(\rho) + \sigma_n^2\mathbf{I} \quad \text{where} \quad \mathbf{K}^{\text{sym}}(\rho) = \begin{pmatrix} \mathbf{K}^p_{PP} & \rho\mathbf{K}^p_{PS} \\ \rho\mathbf{K}^p_{SP} & \mathbf{K}^p_{SS} \end{pmatrix} \tag{15}$$

Asymmetric case

$$\begin{aligned} \mathbf{C}_{\text{asym}}(\rho) &= \mathbf{K}^{\text{asym}}(\rho) + \sigma_n^2\mathbf{I} \\ \text{where } \mathbf{K}^{\text{asym}}(\rho) &= \begin{pmatrix} \mathbf{K}^p_{PP} & \rho\mathbf{K}^p_{PS} \\ \rho\mathbf{K}^p_{SP} & \rho^2\mathbf{K}^p_{SS} + (1-\rho^2)\mathbf{K}^s_{SS} \end{pmatrix} \end{aligned} \tag{16}$$

where we have used the notation $\mathbf{K}^p_{AB}$ to denote the matrix of covariance values, due to k^p, evaluated between $\mathbf{X}_A$ and $\mathbf{X}_B$. In both the symmetric and asymmetric case, the top-left terms in the covariances in the equations (15) and (16) are simply the covariance within the primary task and the cross-terms are due to the assumed correlation of strength ρ between the primary and secondary task. For the asymmetric case, the covariance matrix for the secondary task comes from the "shared" covariance function k^p with the primary task, and a "specific" covariance function k^s. The relationship between the primary and secondary tasks due to the ρ's comes directly from (1) and (5) for the symmetric and asymmetric cases respectively; additionally, the multiplier $(1-\rho^2)$ in the bottom-right term of $\mathbf{K}^{\text{asym}}(\rho)$ corresponds to setting the magnitude of the specific covariance functions to $(1 - \rho^2)$ to achieve a continuum between single-task learning and pooling all tasks.

4.1 Generalisation Error for a Test Point $\mathbf{x}_*$

If the GP prior is correctly specified, then the posterior variance for a new test point $\mathbf{x}_*$ for the primary task (due to the noise free f^p) is also the generalisation error for $\mathbf{x}_*$. The posterior variance at $\mathbf{x}_*$ for the primary task is:

$$\textbf{Symmetric case:} \quad \sigma^2_{\text{sym}}(\mathbf{x}_*, \rho) = k_{**} - \mathbf{k}_*^\top C_{\text{sym}}(\rho)^{-1}\mathbf{k}_* \tag{17}$$

$$\textbf{Asymmetric case:} \quad \sigma^2_{\text{asym}}(\mathbf{x}_*, \rho) = k_{**} - \mathbf{k}_*^\top C_{\text{asym}}(\rho)^{-1}\mathbf{k}_* \tag{18}$$

where k_{**} is the prior variance at $\mathbf{x}_*$, $k^p(\mathbf{x}_*, \mathbf{x}_*)$, and $\mathbf{k}_*^\top = (k^p(\mathbf{x}_*, \mathbf{X}_p) \;\; \rho k^p(\mathbf{x}_*, \mathbf{X}_s))$. We note that the target values y do not affect the posterior variance at the test

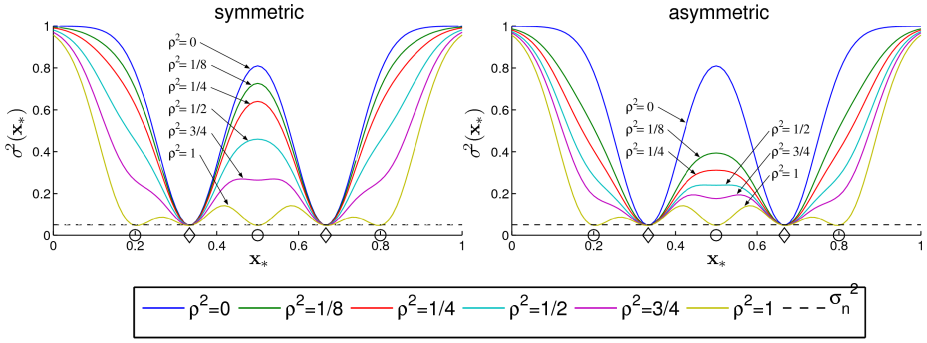


Fig. 2 Behavior of learning in symmetric and asymmetric models. The posterior variances are shown for the test locations $\mathbf{x}_* \in [0, 1]$ given training points from the primary task ($\mathbf{X}_P = [1/3 \;\; 2/3]$, plotted as $\diamond$) and secondary task ($\mathbf{X}_S = [1/5 \;\; 1/2 \;\; 4/5]$, plotted as $\circ$) for the symmetric case (top) and the asymmetric case (bottom). Each plot uses corresponding values of ρ^2 (see legend). The asymmetric learning yields greater reduction of posterior variance at locations of secondary task observations, meaning the primary task is learned better.

locations, and have omitted the dependence on $\mathbf{X}_P$, $\mathbf{X}_S$ and σ_n^2 in the notation for $\sigma^2_{\text{sym}}(\mathbf{x}_*, \rho), \sigma^2_{\text{asym}}(\mathbf{x}_*, \rho)$ for clarity.

To illustrate the difference between the symmetric and asymmetric cases, we plot the posterior variances as a function of $\mathbf{x}_*$ in Figure 2, given two observations for the primary task, and three observations of the secondary task (see figure for more details). Following the setup of Chai (2009), we use a squared exponential covariance function with lengthscale 0.11 for k^p, noise variance $\sigma_n^2 = 0.05$, and, for the asymmetric setup, a squared exponential covariance function with lengthscale 1 for k^s.

Each plot contains 6 curves corresponding to $\rho^2 = [0, 1/8, 1/4, 1/2, 3/4, 1]$, and the dashed line shows the prior noise variance. The training points from the primary task ($\diamond$) create a depression that reaches the prior noise variance for all the curves. However, the depression created by the training points for the secondary task ($\circ$) depends on ρ. For the single task learning case ($\rho = 0$), there is no knowledge transferred from the secondary task. As ρ increases, the generalisation error at the secondary task test points decreases. For the intermediate ρ^2 values (i.e., not 0 or 1 (full correlation)), our asymmetric model gives a smaller posterior variance than the symmetric model at secondary task locations, and therefore suggests better generalisation error.

4.2 Intuition about the Generalisation Errors

Given the illustrative example in the previous section, we sketch the relationship between the generalisation errors for the primary and secondary tasks:

$$\sigma^2_{\text{asym}}(\mathbf{x}_*, \rho) \leq \sigma^2_{\text{sym}}(\mathbf{x}_*, \rho) \tag{19}$$

We show this by considering the covariance matrix at the secondary task points, conditioned on the primary task points. This represents the residual uncertainty

about the secondary task points, given that we know the primary task points. Denoting this quantity as $\mathbf{A}(\rho)$:

$$\mathbf{A}(\rho)_{\text{sym}} = \mathbf{K}^p_{SS} + \sigma_n^2\mathbf{I} - \rho^2\mathbf{K}^p_{SP}(\mathbf{K}^p_{PP} + \sigma_n^2\mathbf{I})^{-1}\mathbf{K}^p_{PS} \quad (20)$$

$$\mathbf{A}(\rho)_{\text{asym}} = \rho^2\mathbf{K}^p_{SS} + (1-\rho^2)\mathbf{K}^s_{SS} + \sigma_n^2\mathbf{I} - \rho^2\mathbf{K}^p_{SP}(\mathbf{K}^p_{PP} + \sigma_n^2\mathbf{I})^{-1}\mathbf{K}^p_{PS} \quad (21)$$

where the multiplier $(1-\rho^2)$ in the asymmetric case is equivalent to setting the overall scale of $\mathbf{K}^s_{SS}$ to $(1-\rho^2)$, which is here done to establish a continuum from single-task learning at $\rho = 0$ and pooling all tasks at $\rho = 1$. If $\mathbf{A}(\rho)_{\text{asym}} \preceq \mathbf{A}(\rho)_{\text{sym}}$ then:

$$\begin{aligned}
\mathbf{A}(\rho)^{-1}_{\text{asym}} &\succeq \mathbf{A}(\rho)^{-1}_{\text{sym}} \\
\mathbf{v}(\rho)^\top\mathbf{A}(\rho)^{-1}_{\text{asym}}\mathbf{v}(\rho) &\geq \mathbf{v}(\rho)^\top\mathbf{A}(\rho)^{-1}_{\text{sym}}\mathbf{v}(\rho) \\
k_{**} - k^p(\mathbf{x}_*, \mathbf{X}_P)(\mathbf{K}^p_{PP} + \sigma_n^2\mathbf{I})^{-1}k^p(\mathbf{x}_*, \mathbf{X}_P) - \mathbf{v}(\rho)^\top\mathbf{A}(\rho)^{-1}_{\text{asym}}\mathbf{v}(\rho) & \\
\leq k_{**} - k^p(\mathbf{x}_*, \mathbf{X}_P)(\mathbf{K}^p_{PP} + \sigma_n^2\mathbf{I})^{-1}k^p(\mathbf{x}_*, \mathbf{X}_P) - \mathbf{v}(\rho)^\top\mathbf{A}(\rho)^{-1}_{\text{sym}}\mathbf{v}(\rho) & \\
\sigma^2_{\text{asym}}(\mathbf{x}_*, \rho) &\leq \sigma^2_{\text{sym}}(\mathbf{x}_*, \rho) \quad (22)
\end{aligned}$$

where we have used the Banachiewicz inversion formula to evaluate the matrix inversions in (17) and (18), and we have defined $\mathbf{v}(\rho) = \rho(k^p(\mathbf{X}_S, \mathbf{x}_*) - \mathbf{K}^p_{SP}(\mathbf{K}^p_{PP} + \sigma_n^2\mathbf{I})^{-1}k^p(\mathbf{X}_P, \mathbf{x}_*))$

The asymmetric model has more flexibility than the symmetric model in the modelling of the secondary task, since it uses both f^p and f^s, rather than just f^p. We expect that $\mathbf{A}(\rho)$ for the asymmetric version would be smaller than for the symmetric since the additional flexibility should allow more accurate modelling of the covariances between the secondary task points, and hence the asymmetric generalisation error should be smaller than the symmetric.

5 Experiments

In this section, we demonstrate the performance of the focused multi-task GP model on a synthetic regression problem in 5.1. In 5.2, we compare our model's performance with alternative models on an asymmetric multi-task classification problem on fMRI data. In all experiments, we use squared exponential covariance functions with automatic relevance determination (ARD) prior: $k(\mathbf{x}, \mathbf{x}') = \sigma_s^2 \exp(-\frac{1}{2}\sum_d (\mathbf{x}_d - \mathbf{x}'_d)^2/l_d^2)$, where σ_s^2 is the overall scale and l_d is the lengthscale for the dth input dimension, initialized to 1. This prior is used for both primary and secondary task functions. With this choice, the hyperparameters to be learned are the lengthscales l_d and overall scale σ_s^2, with separate parameters for the covariance function of the shared components and for each covariance function of each specific component; additionally the parameters of the observation model are learned (noise variance σ_t^2 for each task in a regression setting), and the task similarity coefficients ρ_t.

5.1 Synthetic Data

We use synthetic data to demonstrate how our focused multitask model learns a regression function (the primary task) in conjunction with several related regression problems (the secondary tasks). The model is able to learn the primary

task even where there is missing data, by using the shared signal learned from the secondary tasks. We also show how the number of secondary tasks affects the learning of the primary task: as the number of secondary tasks increases, the mean squared error of the predictions on the test set decreases. In this section we show this behavior in a synthetic experiment where the generation of data follows our model and the ground truth usefulness of the secondary tasks is known; in the next section we will show that good performance is also achieved for asymmetric learning in a real-life case study with several secondary tasks.

Synthetic data is generated as follows (see Fig. 3). All the functions have the same input $\mathbf{x}$, 100 samples evenly spaced on the interval [-5, 5]. The primary task function is generated from $\mathbf{f}^p \sim \mathcal{GP}(0, \mathbf{K}_p)$, where the kernel function is squared exponential with length scale 1 and overall scale $\sigma_s^2 = 1$. The secondary task function in each secondary task s_m is generated according to $\mathbf{f}^{s_m} \sim \mathcal{GP}(\alpha_m \mathbf{f}^p, \beta_m \mathbf{K}_{s_m}^{\text{spec}})$: i.e. the mean is a scaled version (by α_m)of the primary task function. Each specific kernel function $\mathbf{K}_{s_m}^{\text{spec}}$ is squared exponential with lengthscale 1, and α_m is drawn at random from $\mathcal{N}(0, 1)$, β_m at random from $[0, 1]$. We assume a Gaussian observation noise model.

We remove 50 samples from the primary task (see Fig. 4b), and use them as test data. We train the model with different numbers of secondary tasks, ranging from 0 (single task learning) to 24. We repeat the procedure 10 times, randomly drawing the secondary task functions for each run.

Figure 4 (b) shows the mean of the posterior distribution (black) over the primary task function for one of the runs, for different numbers of secondary tasks. We also plot the true underlying primary function (blue line), showing that the model can predict the missing part of the primary task function by transferring information from secondary tasks. The prediction gets nearer to the true underlying primary task function, as the number of relevant secondary tasks increases. Figure 4 (a) shows that the mean squared error on the test set decreases as the number of secondary tasks increases.

5.2 fMRI Data

In this section, we evaluate the performance of our model on fMRI data, obtained from Malinen et al (2007). We consider the task of predicting whether a subject is reacting to a particular stimulus "touch", given the fMRI data. We aim to improve the learning of this primary task by learning it in conjunction with other, related tasks from the other subjects in the experiment. We also include some less related tasks in the secondary task set to show how our model can overcome negative transfer, and focus on the relevant shared signal. The main goal of the experiment is to show that good performance of asymmetric learning can be achieved not only for the artificial data of the previous section but also in a real-life case study, and to moreover show that the asymmetric learning will outperform state of the art alternative methods.

The fMRI data comes from six healthy young adults who participated in two identical sessions, in which they received a continuous 8-min sequence comprising of auditory, visual and tactile stimuli in blocks of 6×33s. The stimuli of different senses never overlapped. Whole-head volumes were acquired with a Signa VH/i

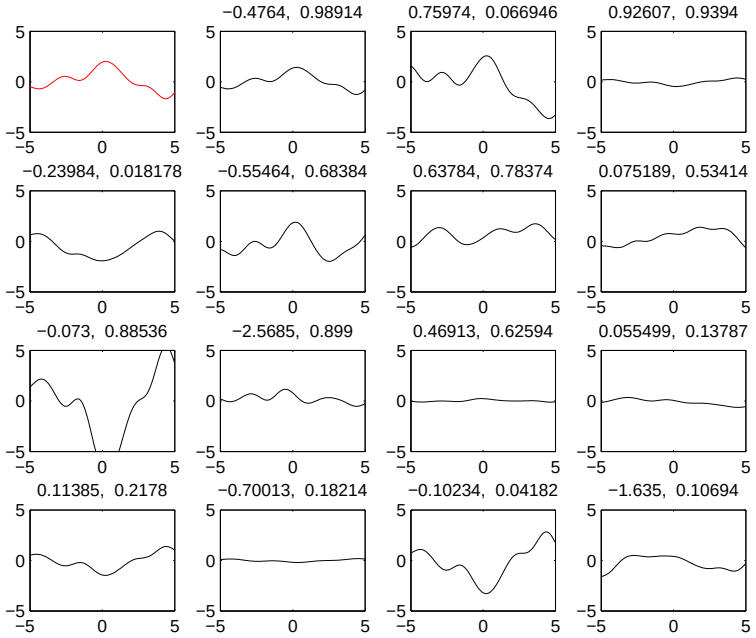
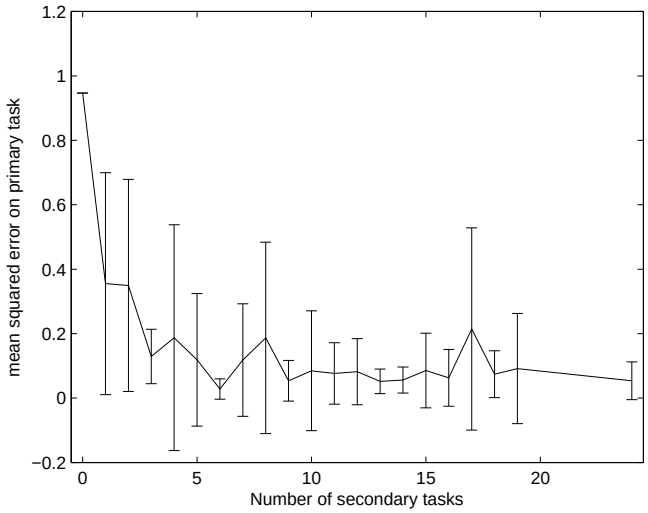


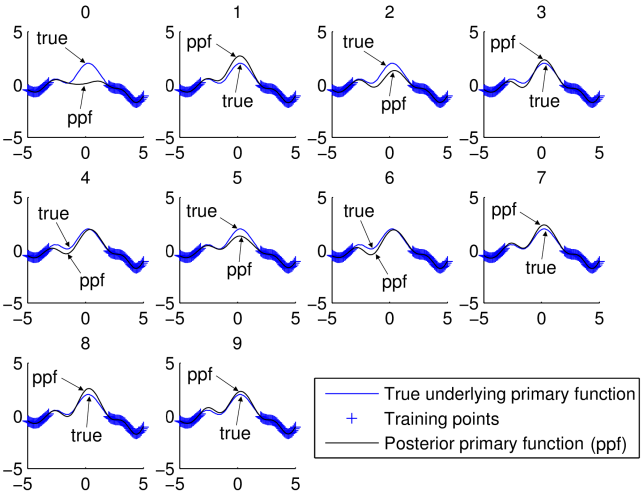
Fig. 3 Synthetic data experiment: experiment setup. We show the functions underlying the generated data: the primary task function (top left, red) and 15 examples of (related) secondary task functions (black). Each function contains a small shared component but also a confounding component of task-specific variation. The weights of the shared and specific functions for the secondary tasks are given above each plot.

3.0 T MRI scanner (General Electric, Milwaukee, WI) using a gradient EPI sequence (TR = 3 s, TE = 32 ms, FOV = 20 cm, flip = 90^o, $64 \times 64 \times 44$ voxels with resolution $3 \times 3 \times 3\text{mm}^3$). In each session, 165 volumes were recorded with the 4 first time points excluded from further analysis. Preprocessing of the fMRI data included realignment, normalization with skull stripping, and smoothing. For additional details on the measurements and applied preprocessing, see Ylipaavalniemi et al (2009). After preprocessing, the dimensionality was reduced to 40 by spatial independent component analysis (ICA) that identified spatial brain activation patterns related to various aspects of the stimuli. For each adult, the resulting data is 161 sets of ICA features (40 dimensional), which can be classified according to one of 6 stimuli ("touch", "auditory" (tones, history, instruction), "visual" (faces, hands, buildings)).

This can be formulated as 6 one-against-all classification tasks in an asymmetric multi-task setup (see Table 1). The classification tasks from subjects 2-4 are similar to the primary task, and may help the learning, whereas 5 and 6 may not be relevant. For each subject the fMRI measurements were done in two separate sessions; in the experiments we use the first session as training data and the second session as test data.



(a)



(b)

Fig. 4 Synthetic data experiment: results of learning with the proposed asymmetric multi-task Gaussian process model. (a) Mean squared error on the primary task test set, over 10 runs, for different numbers of secondary tasks, error bars represent ±1 s.d. (b) Posterior distribution over the primary task function for different numbers of secondary tasks (given above each plot). As the number of secondary tasks grows the learned primary function becomes close to the true underlying function and the mean squared error decreases.

Table 1 Asymmetrical multi-task set up for fMRI data study

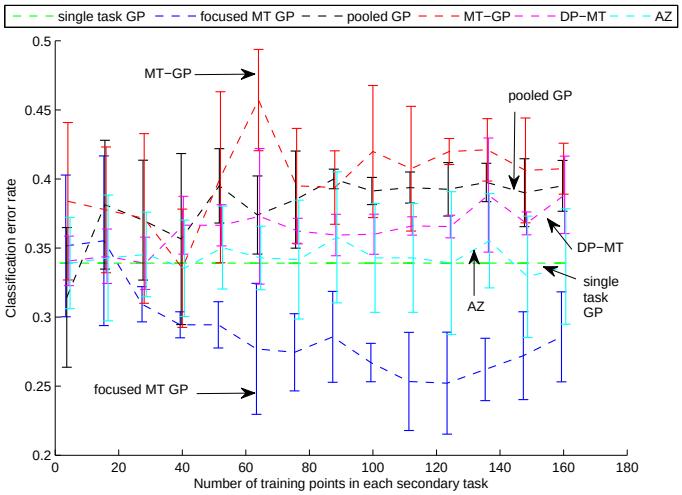
Subject	Classification Task
1 (primary)	"touch" against all
2 (secondary)	"touch" against all
3 (secondary)	"touch" against all
4 (secondary)	"touch" against all
5 (secondary)	"auditory" (instruction) against all
6 (secondary)	"visual" (buildings) against all

We compare the focused multi-task learning approach ("focused MT-GP") with five reference models. The first baseline model is single task learning using GP classification ("single task GP"), trained only on the samples of the primary task. The second ("pooled GP") learns a GP classification model from the training examples from all tasks (i.e. treating all data as a single task). For "pooled GP" we use a sparse approximation when the number of training examples > 300, using 30 pseudo-inputs. We also compare to three state-of-the-art methods, one transfer learning method and two (symmetric) multi-task learning methods: the predictive structure learning method of Ando and Zhang (2005, "AZ"), the symmetric multi-task learning with Dirichlet process priors method ("DP-MT") from Xue et al (2007), and the symmetric multi-task GP method ("MT-GP") from Bonilla et al (2008); the symmetric multi-task GP method was previously discussed in Sections 2.1 and 2.2. For the "AZ" method, we fix the dimension of the shared predictive structure heuristically to $h = 26$, after performing PCA across all the training samples (primary and secondary) and find the dimension of the subspace that explains 80% of the variance.

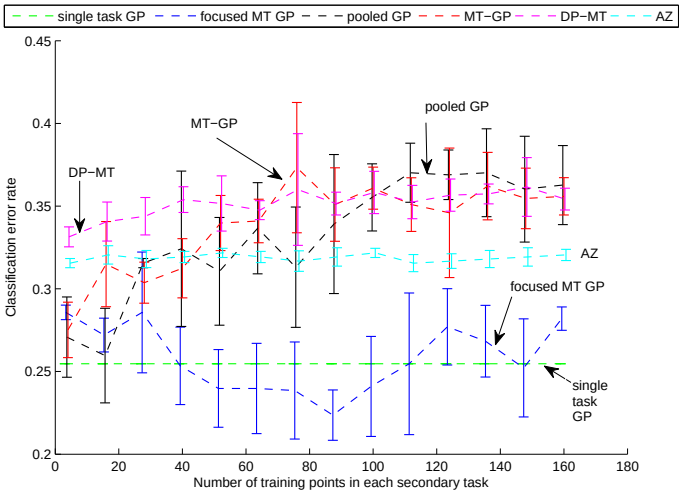
We evaluate the methods using a fixed number of training examples in the primary task (64 and 161), while varying the number of training examples in each secondary task (ranging from 4 to 160), over 5 repetitions. We change the amount of secondary task data to investigate how the models' performance is affected by the tasks (2-4) that may help learning on the primary task, and the more unrelated tasks (5-6). Note that the number of secondary tasks is fixed, only the amount of data in the secondary tasks changes. Due to the class imbalance in the data, when randomly picking a subset of secondary training task examples, we ensure that there is at least one positive and one negative example. For the GP-based methods, we also fix the bias parameter $b = \Phi^{-1}(r)$ of the probit noise model, where r is the ratio of positive samples to negative samples in the training data.

Figure 5 displays the classification error on the test set for the primary task, over different numbers of training examples for the secondary tasks, for 64 training examples in the primary task (a) and 161 (*i.e.*, all available training examples for the primary task) in (b).

Pooling of samples seems to always be a bad choice on this data. We also find that the symmetric models (MT-GP and DP-MT) perform poorly: both work only roughly equally to single-task learning for small numbers of secondary task data and the performance worsens as amount of secondary data increases. Hence it seems that the secondary data here differs from primary data to the extent of causing negative transfer. AZ seems to work better but at most on the same level as single task learning. More work would be needed for model selection, however, which might improve performance.



Number of primary task training examples = 64



Number of primary task training examples = 161

Fig. 5 Classification error on test set for the primary task, against the number of training examples in each secondary task for different primary task training set sizes (top subfigure: small, bottom subfigure: larger). Our focused multi-task learning approach "Focused MT-GP" outperforms the other methods especially when there are few primary task samples and a reasonable amount of secondary-task samples. When the number of primary-task samples is larger, single-task learning also performs well.

Focused MT-GP seems able to leverage on the secondary tasks, clearly outperfoming others including single task learning when the amount of data in the primary task is small. Multitask learning is most relevant when the primary task has little data; Focused MT-GP performs well in this scenario. When there is more primary data single task learning improves rapidly, although in Figure 5 Focused MT-GP still outperforms it. Focused MT-GP seems to need more than a few samples in the secondary tasks in order to perform well; the explanation is probably that for this data it is hard to distinguish between useful and negative transfer, and more data is needed to make the choice. Bad performance of pooling and symmetric multi-task approaches supports this interpretation. We will investigate the effect of small sample sizes on negative transfer in future work; the current result already shows that the asymmetric learning works well and outperforms other methods given a reasonable number of samples in secondary tasks.

6 Investigating effect of negative transfer in our model

Negative transfer essentially happens when a model mistakes non-related properties of a secondary task as being related. Although this might happen with small sample sizes even in a well-specified model, it may become much more prevalent if the model assumptions are incorrect. Although our asymmetric learning model involves flexible assumptions about task relationships, it is important to examine how well the model performs when the assumptions are violated. In this section we study the effect of violating the model assumptions, and the resulting negative transfer, in a controlled setting.

Our model is based on the assumption that there is an asymmetrical sharing structure within the data, with the emphasis on learning the sharing between the primary and secondary tasks. However, if there is strong shared structure between the secondary tasks which is not shared with the primary task, this could cause the model to learn that shared structure rather than the primary task function, yielding negative transfer to the primary task.

We demonstrate this effect with a toy data experiment. Synthetic data is generated as follows: The primary task function is generated from $\mathbf{f}^p \sim \mathcal{GP}(0, \mathbf{K}_p)$, where the kernel function is squared exponential with length scale 0.5. This is divided into training and test inputs. A "shared noise function" f_a, which is shared between secondary tasks only, is generated from a GP with squared exponential function and lengthscale (1/3). The secondary task functions for the secondary tasks s_m are generated according to $\mathbf{f}^{s_m} \sim \mathcal{GP}((s-1)\alpha_{m,1}\mathbf{f}_p + s\alpha_{m,2}\mathbf{f}_a, \beta_m \mathbf{K}_{s_m}^{\text{spec}})$ where $\mathbf{f}_a$ are the values generated for the shared noise function. Each specific kernel function is squared exponential with lengthscale 0.5. The $\alpha_{m,1}, \alpha_{m,2}, \beta_m$ are drawn uniformly at random from $[0, 1]$. The s is an indicator function to show whether the secondary task shares f_a. We also add Gaussian noise generated from $\mathcal{N}(0, 0.01)$. We generate 10 secondary task functions, and vary the number of secondary tasks that share f_a (the number which have $s = 1$) from 0 to 10, and use 10 replications. Figure 6(a) shows the mean squared error between the true underlying function and the predictive posterior mean over the test inputs, for each value of s; the mean squared error remains low up to 6 tasks with the shared noise function. (b) shows the correlation coefficient ρ averaged over the secondary tasks, as the strength of f_a increases; the average correlation coefficient decreases when

ever more tasks use the shared noise function, showing that the model correctly learns that many of the secondary tasks are not useful for the primary task. Figure 6(c-f) shows the posterior means for the test inputs for each run.

Overall, in this experiment our method appears reasonably tolerant against the presence of the non-useful shared noise function, maintaining good performance as the number of tasks featuring that function rises: as shown in Figure 6(a), performance remains stable even up to 6 tasks featuring the shared noise function.

7 Asymmetric vs. symmetric multi-task learning

In the fMRI case study of Section 5.2 our asymmetric multi-task learning method outperformed several comparison methods, including the most closely related symmetric multi-task learning approach, the method of Bonilla et al (2008) which is based on Gaussian processes and is here called "Symmetric multi-task GP". Unlike our method, Symmetric MT-GP treats all tasks as equally important. The mathematical formulation of Symmetric MT-GP has been briefly discussed in Sections 2.1 and 2.2. In this section we show that both our asymmetric model and the symmetric model will perform well for certain domains of problems, and both should be part of the multi-task learning "toolbox".

We compare the performance of our method and the symmetric MT-GP on a continuum of multi-task learning problem domains. At the left end of the continuum, the problems follow the assumptions of our focused multi-task learning GP ("focused MT-GP"), and at the right end the problems follow the assumptions of symmetric MT-GP. For each domain, and each learning problem in the domain, the performance of the methods is evaluated by mean-square error over test samples in the primary task.

In detail, we evaluate the performance of the methods at 10 points along the continuum of domains, and we generate 30 multi-task learning problems from each domain along the continuum. All the learning problems are regression problems similar to Section 5.1: each problem contains 10 one-dimensional regression tasks (data sets), where the first task is the primary task and others are secondary tasks. Each secondary task has 50 input samples uniformly distributed along the interval $[0, 1]$. The primary task has fewer samples, and moreover all primary task samples in the middle interval $[0.25, 0.75]$ have been left out of the training data, leaving 15 samples on average in the primary task. This design was chosen to highlight the multi-task learning ability of the methods: because training samples in the primary task are not provided for the middle of the input interval in the primary task, the primary function along the middle interval can only be learned well by learning across tasks.

In each task, the outputs for the inputs are generated from a weighted sum of Gaussian process functions, plus observation noise from a Gaussian observation noise model. The weighting of the GP functions is generated according to the domain: at the left end of the domain continuum the functions follow the Focused multi-task GP model, so that the primary task uses a single GP function $\mathbf{f}^p \sim \mathcal{GP}(0, \mathbf{K}_p)$, and in each secondary task s_m the GP function $f^{s_m} = \alpha_m \mathbf{f}^p + \mathbf{f}^{s_m,\text{specific}}$ is a sum of the primary function and and a specific function $f^{s_m,\text{specific}} \sim \mathcal{GP}(0, \mathbf{K}_{s_m}^{\text{spec}})$, where the multiplier α_m is drawn uniformly from $[0, 1]$. All kernels $\mathbf{K}_p$ and $\mathbf{K}_{s_m}^{\text{spec}}$ are squared exponential kernels with length scale

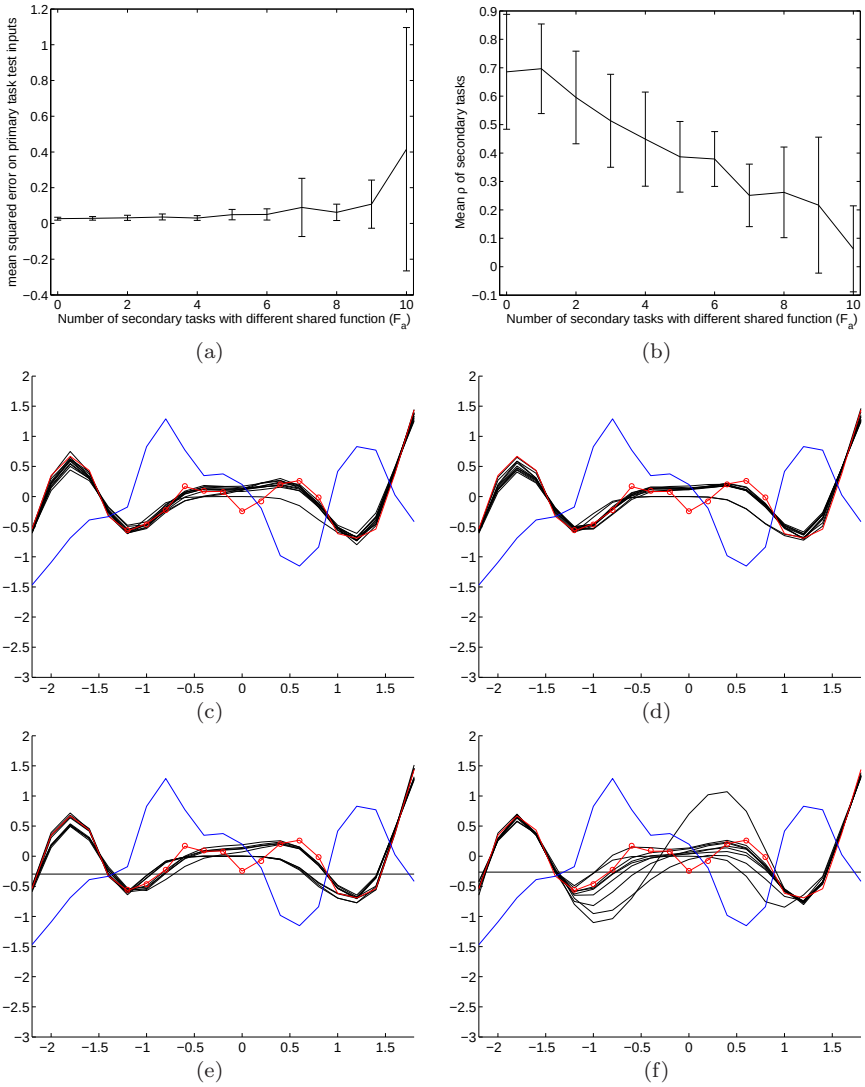


Fig. 6 Synthetic data experiment: effect of sharing between secondary tasks on the proposed asymmetric multi-task Gaussian process model. (a) Mean squared error on the primary task test set, over 10 runs, for different numbers of shared secondary tasks (with $s = 1$), error bars representing ± 2 s.d. (b) Mean correlation coefficient ρ over secondary tasks, over 10 runs, for different numbers of shared secondary tasks (with $s = 1$), error bars representing ± 2 s.d. (c)-(f) Posterior distribution over the primary task function (black lines) at test points for different numbers of secondary tasks ($0, 2, 5, 9$ respectively) sharing an additional function (f_a). True values of test points are given by red circles, true value of primary task function f_p as red line, f_a as blue line. As the number of secondary tasks that have a shared additional function grows, the average of correlation coeffients learned by our model decreases as expected. Our model performs well for a reasonable number of such secondary tasks, up to about 6 such tasks; the prediction error on the primary task remains low.

0.05. At the right end of the continuum, 10 GP functions are shared across all tasks, so that $\mathbf{f}_k \sim \mathcal{GP}(0, \mathbf{K}_k)$ for $k = 1, \ldots, 10$, and each primary or secondary task uses four of the 10 functions with random weights: $f_p = \sum_{k=1}^{K} w_{p,k} f_k$ and $f_p = \sum_{k=1}^{K} w_{s,k}^m f_k$ where a randomly chosen subset of four weights $w_{p,k}$ are drawn uniformly from $[0, 1]$ and the other six weights are zero, and similarly for weights $w_{p,k}^m$ in each secondary task. This generative process at the right end of the continuum follows the assumptions of the Symmetric multi-task GP. In the intermediate domains the weights are generated with both procedures and linearly mixed together, yielding a smooth transition along the continuum from one kind of learning problems to the other. We ran both our Focused multi-task GP method and the Symmetric multi-task GP method for all problems in all domains and computed the error on test data from the primary task of each problem.[1]

The results are shown in Figure 7. Our Focused multi-task GP performs as desired: in domains near the left end of the continuum which are reasonably close to our assumptions the asymmetric multi-task learning approach outperforms the symmetric approach, whereas near the right end of the continuum the symmetric approach performs better. This shows that both symmetric and asymmetric learning should be part of the "toolbox" for multi-task learning scenarios, and analysts should consider whether each application is likely to benefit from the asymmetric or symmetric approach.

8 Model with Several Shared Components

In our asymmetric learning model, only one shared function was used between tasks; this was already sufficient to yield very good performance in the previous experiments. We now point out that our model is not limited to one shared function: it is easy to extend our Focused multi-task Gaussian process model to incorporate more than one shared function that contribute to the primary task and which can be shared differently among different secondary tasks.

Note that in the extended model we will present in this section, there is still only one primary task; the difference is that the model can now handle more complex sharing between the primary task and the secondary tasks since the component functions of the primary task can each be shared differently with the secondary tasks.

We again define the assumed functional relationships between tasks and then derive the corresponding multi-task GP prior. With L shared functions $f^{p,l}$, $l = 1, \ldots, L$, the primary task GP function is a simple sum of the shared functions: $f^p = \sum_{l=1}^{L} f^{p,l}$. In each secondary task s_i, the GP function is a weighted sum of the shared functions plus a task-specific function: $f^{s_i} = f^{s_i,\text{specific}} + \sum_{l=1}^{L} \rho^{s_i,l} f^{p,l}$.

[1] For Symmetric multi-task GP we used its authors' implementation available at http://users.cecs.anu.edu.au/∼u4882938/code.html, with random initializations of the multitask matrix and fixed initialization for other parameters. For our method, we did not use the computational speedup approximation of equation (9) since the data sets are fairly small, and we placed simple flat priors for the hyperparameters of the GPs. Both methods were run in Matlab and we kept their running times roughly equal (221s per problem for our method, 279s per problem for the symmetric multi-task GP); in that time, we were able to run three runs per problem of Focused multi-task GP from random initializations, taking the run with the best internal cost function value, and one run per problem of Symmetric multi-task GP.

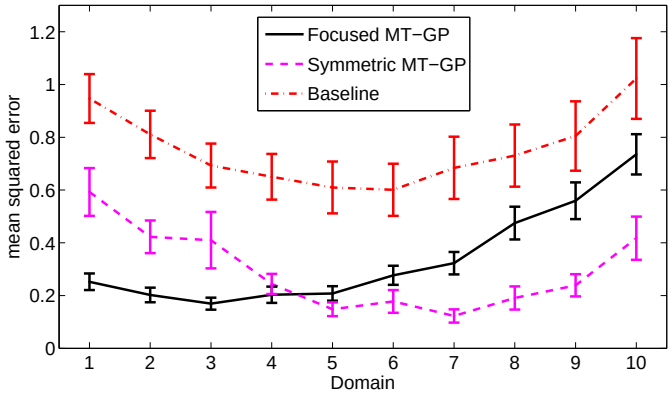


Fig. 7 Comparison of focused (asymmetric) and symmetric Gaussian process based multi-task learning on a continuum of multi-task problem domains. The comparison methods are our focused multi-task GP method ("Focused MT-GP") and a symmetric multi-task GP method (Bonilla et al, 2008) denoted "Symmetric MT-GP", and "Baseline" denotes simply predicting zero. At left in the continuum, learning problems follow the assumptions of our focused multi-task GP method and at right they follow the assumptions of the symmetric multi-task GP method (Bonilla et al, 2008). For each method, the line shows the mean squared prediction error (smaller numbers are better) over 30 learning problems from each domain and over the test data within each problem; error bars show the standard deviation (over learning problems) of the mean. The asymmetric and symmetric GP methods both perform well towards different ends of the continuum as desired.

Note that each secondary task uses each shared function with a different multiplier $\rho^{s_i,l}$, allowing different secondary tasks to share different kinds of shared functions, and the variation not shared with the primary task is again explained away with the task specific GP function $f^{s_i,\text{specific}}$.[2] Figure 8 shows the setup. The corresponding covariance function between outputs $y_j^{s_i}$ and $y_{j'}^{s_{i'}}$ in two secondary tasks s_i and $s_{i'}$ is

$$\left\langle y_j^{s_i}, y_{j'}^{s_{i'}} \right\rangle = \left(\delta(s_i, s_{i'}) + \sum_{l=1}^{L} \rho^{s_i,l}\rho^{s_{i'},l}\right) k^p(x_j^{s_i}, x_{j'}^{s_{i'}}) \tag{23}$$

where the function δ is 1 if the task indices s_i and $s_{i'}$ are the same and zero otherwise; the covariance between a secondary task and the primary task (say s_i is the primary task) is the same except that the multipliers $\rho^{s_i,l}$ are replaced by ones and the function δ is replaced by zero; and the covariance within the primary task is simply L times the covariance function k^p over inputs. For simplicity we did not apply the computational speedup approximation used in (9) here, so equation (23) is directly used to compute the coveriance function. This covariance function defines the GP prior over all tasks; Bayesian inference over the functions then proceeds as before, and the hyperparameters of the GP prior can again be learned by maximizing marginal log likelihood. Our model with a single shared component is a special case of this model with $L = 1$.

[2] Note that there is no need to consider more than one specific function for a secondary task: if several GP functions specific to some secondary task exist, their total contribution to the secondary task function is a weighted sum equivalent to a single specific GP function.

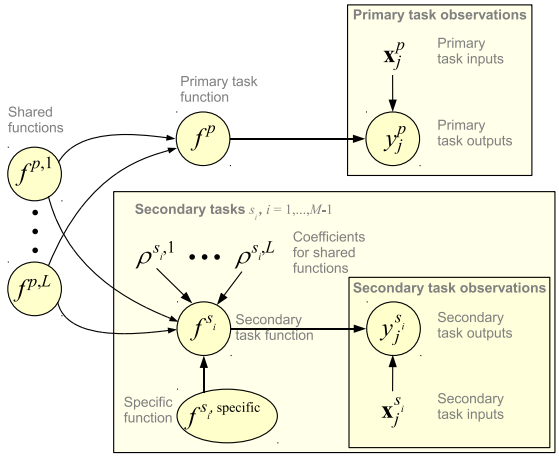


Fig. 8 Graphical model of the focused multi-task Gaussian process model with multiple shared components, showing the relationship between values of the primary task and secondary tasks; each secondary task can use each of the shared functions with different strengths. The parameters of the covariance functions have been omitted for clarity. Compare to Figure 1.

We briefly test this extended model on a multi-task problem with 15 one-dimensional regression tasks: the 14 secondary tasks have 100 samples each in $[0, 1]$ and the primary task has 52 samples in $[0, 1]$ excluding the middle interval $[0.25, 0.75]$. The primary task function to be learned is a sum of two sinusoid components $\sin(7x)$ and $0.8\cos(30x)$; the secondary tasks each share the two sinusoids with different strenghts and also contain a specific function generated from a Gaussian process; a small amount of Gaussian observation noise is added to data of each task. The training data are shown in Figure 9(left). We use the extended version of our model, here with two shared functions, to learn the primary task well. The resulting prediction is shown in Figure 9(right) and yields mean squared error 0.005 over 101 equally-spaced test points, whereas a model learned with only one shared function would yield a larger mean squared error 0.022.

In our fMRI case study the model with a single shared component already yielded good results, but as shown here the flexibility provided by multiple shared components may be useful in future application domains.

9 Conclusion

We derived a multi-task Gaussian process learning method, the "focused multi-task GP", designed for asymmetrical multi-task learning scenarios, to facilitate improved learning on a primary task through the transfer of relevant information from a set of potentially related secondary tasks. The novel dependency structure was formulated based on the GP predictive distribution over the secondary tasks given the primary task, and constraining the secondary tasks to be conditionally independent. After observing the primary task, the primary task function can be used to predict a part of each secondary task, depending on the degree of task relatedness, which is learned during the optimisation. The model also permits each secondary task to have its own task-specific variation which is unshared with

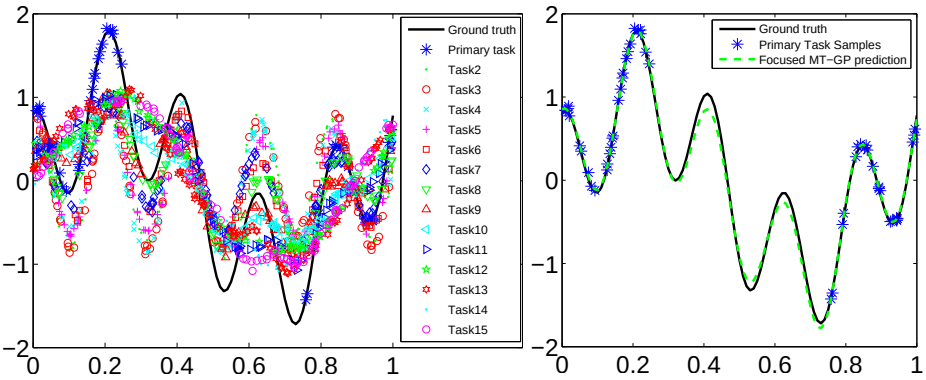


Fig. 9 Example of focused multi-task learning with more than one shared component (here two shared components). Left: a multi-task data set; the primary function is a sum of two sinusoids, and each secondary task may share the sinusoids with different strenghts. Right: prediction from the Focused multi-task Gaussian process model ("Focused MT-GP") learned with two shared components. The model performs well, yielding mean squared error 0.005.

the primary task, and this flexibility should cause the model to focus on modelling the primary task function well. We demonstrated the model on synthetic data and an asymmetrical multi-task learning problem with fMRI data, and showed improved performance over baseline approaches, and a state of the art transfer learning and multi-task learning method. We also experimentally demonstrated the performance of the model with increasing non-useful shared variation. We demonstrated that the model outperforms the comparable symmetric multi-task approach over several problem domains, and overall showed that both symmetric and asymmetric models should be part of the multi-task learning "toolbox". Lastly we presented an extension of the model to several components shared with the primary task, and demonstrated its good performance in an initial experiment.

The key idea in the model is to make simplifying conditional independence assumptions about the relationships of the secondary tasks, but compensate the simplicity by adding a flexible "explaining away" model for each secondary task to reduce negative transfer. This structure is expected to perform well when the data fulfills the independence assumptions, but additionally due to the "explaining away" models reasonably well also in the ubiquitous case where the data does not exactly fit either this model or its alternatives. The performance was demonstrated empirically in this paper, and also analyzed briefly. More theoretical analysis of the power of the "explaining away" models is still needed.

Acknowledgements The authors belonged to the Adaptive Informatics Research Centre, a national CoE of the Academy of Finland, and SK and JP currently belong to the Finnish Centre of Excellence in Computational Inference Research COIN (grant no 251170). JP was supported by the Academy of Finland, decision numbers 123983 and 252845. This work was also supported in part by the PASCAL2 Network of Excellence, ICT 216886, and by the Tekes Multibio project.

References

Alvarez M, Lawrence ND (2009) Sparse convolved Gaussian processes for multioutput regression. In: Koller D, Schuurmans D, Bengio Y, Bottou L (eds) Advances in Neural Information Processing Systems 21, MIT Press, Cambridge, MA, pp 57–64

Ando RK, Zhang T (2005) A framework for learning predictive structures from multiple tasks and unlabeled data. Journal of Machine Learning Research 6:1817–1853

Bickel S, Bogojeska J, Lengauer T, Scheffer T (2008) Multi-task learning for HIV therapy screening. In: McCallum A, Roweis S (eds) Proceedings of the 25th Annual International Conference on Machine Learning (ICML 2008), Omnipress, pp 56–63

Bickel S, Sawade C, Scheffer T (2009) Transfer learning by distribution matching for targeted advertising. In: Koller D, Schuurmans D, Bengio Y, Bottou L (eds) Advances in Neural Information Processing Systems 21, MIT Press, Cambridge, MA, pp 145–152

Bonilla EV, Chai KMA, Williams CKI (2008) Multi-task Gaussian Process Prediction. In: Platt JC, Koller D, Singer Y, Roweis S (eds) Advances in Neural Information Processing Systems 20, MIT Press, Cambridge, MA, pp 153–160

Caruana R (1997) Multitask learning. Machine Learning 28:41–75

Chai KMA (2009) Generalization errors and learning curves for regression with multi-task gaussian processes. In: Bengio Y, Schuurmans D, Lafferty J, Williams CKI, Culotta A (eds) Advances in Neural Information Processing Systems 22, MIT Press, Cambridge, MA, pp 279–287

Malinen S, Hlushchuk Y, Hari R (2007) Towards natural stimulation in fMRI - issues of data analysis. Neuroimage 35(1):131– 139

Marx Z, Rosenstein MT, Kaelbling LP (2005) Transfer learning with an ensemble of background tasks. In: Inductive Transfer: 10 Years Later, NIPS 2005 workshop

Minka T (2001) Expectation Propagation for approximative Bayesian inference. In: Breese JS, Koller D (eds) Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence, Morgan Kaufmann Publishers, San Fransisco, CA, pp 362–369

Pan SJ, Yang Q (2010) A survey on transfer learning. IEEE Transactions on Knowledge and Data Engineering 22:1345–1359

Raina R, Ng AY, Koller D (2005) Transfer learning by constructing informative priors. In: Inductive Transfer: 10 Years Later, NIPS 2005 workshop

Rosenstein MT, Marx Z, Kaelbling LP (2005) To transfer or not to transfer. In: Inductive Transfer: 10 Years Later, NIPS 2005 workshop

Snelson E, Ghahramani Z (2006) Sparse Gaussian Processes using Pseudo-inputs. In: Weiss Y, Schölkopf B, Platt J (eds) Advances in Neural Information Processing Systems 18, MIT Press, Cambridge, MA, pp 1257–1264

Teh YW, Seeger M, Jordan MI (2005) Semiparametric latent factor models. In: Cowell RG, Ghahramani Z (eds) Proceedings of AISTATS 2005, the Tenth International Workshop on Artificial Intelligence and Statistics, Society for Artificial Intelligence and Statistics, pp 333–340, (Available electronically at http://www.gatsby.ucl.ac.uk/aistats/)

Teh YW, Jordan MI, Beal MJ, Blei DM (2006) Hierarchical dirichlet processes. Journal of the American Statistical Association 101(476):1566–1581

Thrun S (1996) Is learning the n-th thing any easier than learning the first? In: Touretzky DS, Mozer MC, Hasselmo ME (eds) Advances in Neural Information Processing Systems 8, MIT Press, Cambridge, MA, pp 640–646

Wackernagel H (1994) Cokriging versus kriging in regionalized multivariate data analysis. Geoderma 62:83 – 92

Wu P, Dietterich TG (2004) Improving SVM accuracy by training on auxiliary data sources. In: Greiner R, Schuurmans D (eds) Proceedings of the 21st International Conference on Machine Learning (ICML 2004), Omnipress, Madison, WI, pp 871–878

Xue Y, Liao X, Carin L (2007) Multi-Task Learning for Classification with Dirichlet Process Priors. Journal of Machine Learning Research 8:35–63

Ylipaavalniemi J, Savia E, Malinen S, Hari R, Vigário R, Kaski S (2009) Dependencies between stimuli and spatially independent fMRI sources: Towards brain correlates of natural stimuli. Neuroimage 48:176–185

Yu K, Tresp V (2005) Learning to learn and collaborative filtering. In: Inductive Transfer: 10 Years Later, NIPS 2005 workshop

Yu K, Chu W, Yu S, Tresp V, Z X (2007) Stochastic Relational Models for Discriminative Link Prediction. In: Schölkopf B, Platt J, Hoffman T (eds) Advances in Neural Information Processing Systems 19, MIT Press, Cambridge, MA, pp 1553–1560