Accelerated Bayesian Optimization through Weight-Prior Tuning

Alistair Shilton[1] **Sunil Gupta**[1] **Santu Rana**[1]
Pratibha Vellanki[2] **Laurence Park**[3]
Cheng Li[4] **Svetha Venkatesh**[1] **Thomas Dorin**[5]
Alessandra Sutti[5] **David Rubin**[5]
Teo Slezak[5] **Alireza Vahid**[5] **Murray Height**[5]

[1]Applied Artificial Intelligence Institute (A^2I^2), Deakin University, Geelong, Australia
[2]Office of National Statistics, Southampton, United Kingdom
[3]School of Computing, Engineering and Mathematics, University of Western Sydney, Sydney, Australia
[4]School of Computing, National University of Singapore, Singapore
[5]Institute for Frontier Materials (IFM), Deakin University, Geelong, Australia

Abstract

Bayesian optimization (BO) is a widely-used method for optimizing expensive (to evaluate) problems. At the core of most BO methods is the modeling of the objective function using a Gaussian Process (GP) whose covariance is selected from a set of standard covariance functions. From a weight-space view, this models the objective as a linear function in a feature space implied by the given covariance K, with an arbitrary Gaussian weight prior $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. In many practical applications there is data available that has a similar (covariance) structure to the objective, but which, having different form, cannot be used directly in standard transfer learning. In this paper we show how such auxiliary data may be used to construct a GP covariance corresponding to a more appropriate weight prior for the objective function. Building on this, we show that we may accelerate BO by modeling the objective function using this (learned) weight prior, which we demonstrate on both test functions and a practical application to short-polymer fibre manufacture.

1 Introduction

Bayesian Optimization (BO) [Snoek et al., 2012, Brochu et al., 2010] is a form of sequential model-based optimization (SMBO) that aims to to find $\mathbf{x}^* = \text{argmax}_{\mathbf{x}} f(\mathbf{x})$ with the least number of evaluations for an expensive (to evaluate) function $f : \mathbb{X} \to \mathbb{R}$. In many cases there exists an additional, auxiliary dataset $\mathtt{A}$ relating to f that is relevant to the problem at hand - for example patents and technical handbooks that contain condensed knowledge related to f, data obtained from similar (possibly older or superseded) equipment and, more generally, results from optimizing structurally similar functions. It is not necessary that the auxiliary data be generated by f (if $\mathtt{A}$ was generated by a function sufficiently similar to f then we could simply use standard transfer learning techniques [Pan and Yang, 2010, Yogatama and Mann, 2014, Bardenet et al., 2013, Joy et al., 2016, Shilton et al., 2017]); rather, what matters is that the auxiliary data is structurally similar to f insofar as it has a similar *covariance* structure. Human experimenters routinely use such data to inform and accelerate the (experimental) optimization process. In the present paper we investigate the use of such data to accelerate Bayesian Optimization.

Typically in BO it is assumed that $f \sim \mathcal{GP}(0, \tilde{K})$ is a draw from a zero mean Gaussian Process (GP) [Rasmussen, 2006] with covariance $\tilde{K} : \mathbb{X} \times \mathbb{X} \to \mathbb{R}$. In general $\tilde{K}$ is unknown, so we select an alternative K to use in its place. When selecting K we use intuitions and experimentalist knowledge relating to f - for example how smooth is f (should we use squared exponential or Matérn covariance?), is the length-scale consistent everywhere (should be use an isotropic or anisotropic covariance?) etc, but typically we restrict our choice to a subset of standard kernels. This introduces approximations and experimenter biases, as our knowledge of the covariance structure of f is by definition incomplete in most cases, and moreover it appears unlikely that the covariance structure can captured by a small set of generic "standard" covariance functions. Subsequently BO is likely to converge more slowly than it should as the GP used to model f has an inaccurate covariance function.

To overcome the deficiencies it is standard practice to tune hyper-parameters of K at each iteration of the BO, ei-

Proceedings of the 23[rd]International Conference on Artificial Intelligence and Statistics (AISTATS) 2020, Palermo, Italy. PMLR: Volume 108.

ther to maximize the log-likelihood function or minimize some sort of approximation error (e.g. leave-one-out error). The auxiliary dataset may be used to accelerate the process by pre-tuning hyper-parameters or, if it is large enough, techniques such as multi-kernel learning (MKL) [Lanckriet et al., 2004, Bach et al., 2004] may be used to find to find a more nuanced fit. Nevertheless there remains an underlying assumption that the space spanned by the chosen subset of standard kernels contains a good approximation of the actual covariance.

In this paper we provide a principled alternative approach to covariance construction using auxiliary data. Starting with the weight-space perspective of GPs [Rasmussen and Williams, 2006], our approach pre-selects a weight prior by modeling the auxiliary data using kernel methods and then transfers this to the BO problem using m-kernel theory. From the weight-space perspective a GP models $f \sim \mathcal{GP}(0, K)$ as a linear function in feature space $f(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\boldsymbol{\varphi}(\mathbf{x})$ with a weight prior $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ [Rasmussen and Williams, 2006], where the feature map $\boldsymbol{\varphi} : \mathbb{X} \to \mathbb{R}^d$ is implicitly defined by the covariance (kernel) K via Mercer's condition [Mercer, 1909]. Thus when we use a standard covariance K we are stating a belief that:

1. f is linear in the weight space defined by K.
2. The features $\varphi_i(\mathbf{x})$ are uncorrelated and equally important in this approximation (hence $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$).

For universal kernels [Micchelli et al., 2006, Sriperumbudur et al., 2011] such as the squared-exponential (SE) kernel or the Matérn kernels the first assumption will be true for most (reasonably well-behaved) functions f, as such kernels can approximate most (reasonably well-behaved) functions f to arbitrary accuracy. By contrast, the second assumption is much more tenuous. It will almost certainly be incorrect if the kernel hyper-parameters are incorrectly specified, and there is no a-priori reason to believe it will be accurate in general. This motivates us to replace the weight prior with the more general weight prior $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\boldsymbol{\tau}})$, where the covariance $\boldsymbol{\Sigma}_{\boldsymbol{\tau}}$ is a diagonal matrix learned from the auxiliary data. This embodies the alternative belief:

2 (alt). The features $\varphi_i(\mathbf{x})$ are uncorrelated *and their relative importance may inferred from the auxiliary dataset* $\mathtt{A}$ (hence $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\boldsymbol{\tau}})$).

To achieve our goal requires two steps, namely (1) a heuristic to extract relative feature importance information from the auxiliary dataset $\mathtt{A}$, and (2) a way to alter the weight-prior of a GP to reflect the relative feature importance information found in step (1).

To address the first goal we we use standard machine (kernel) learning techniques. If we apply a support vector machine (SVM) [Cortes and Vapnik, 1995] method (or similar) equipped with kernel K to the auxiliary dataset $\mathtt{A}$ then the answer we obtain takes the form of the representation $\alpha_0, \alpha_1, \ldots \in \mathbb{R}$ of a weight vector $\tilde{\mathbf{w}} = \sum_i \alpha_i \boldsymbol{\varphi}(\mathbf{x}_i)$ in feature space. Assuming a reasonable "fit" we hypothesize that the weights $\tilde{w}_i$ will be larger in magnitude for relevant features and smaller for irrelevant ones. Thus we generate (implicit) information about feature relevance for $\mathtt{A}$.

To address the second goal we borrow from ℓ^p-norm regularization [Salzo et al., 2018, Salzo and Suykens, 2016] and large-margin L^p-moment classifiers[1] [Der and Lee, 2007], and particularly m-kernels (aka tensor kernels [Salzo and Suykens, 2016] or moment functions [Der and Lee, 2007]), which are an extension of kernels to multi-linear products in feature space. We introduce a class of m-kernel families, the free kernels, that are expandable as weighted multi-linear products in feature space - that is, families of functions $K : \mathbb{X}^m \to \mathbb{R}$ for which there exists $\boldsymbol{\vartheta} : \mathbb{X} \to \mathbb{R}^d$, $\boldsymbol{\tau} \in \mathbb{R}^d$ such that:

$$K_m(\mathbf{x}, \mathbf{x}', \ldots, \mathbf{x}'''') = \textstyle\sum_j \tau_j^2 \vartheta_j(\mathbf{x}) \vartheta_j(\mathbf{x}') \ldots \vartheta_j(\mathbf{x}'''')$$

where we note that many free kernels share the same unweighted feature map $\boldsymbol{\vartheta}$. By reformulating Gaussian Processes in terms of free kernels we demonstrate that the weights $\boldsymbol{\tau}$ play the role of (diagonal) weight prior covariances $\boldsymbol{\Sigma}_{\boldsymbol{\tau}} = \mathrm{diag}(\boldsymbol{\tau}^{\odot 2})$. Finally, we show how this weight $\boldsymbol{\tau}$ may be changed to match the weight extracted from the auxiliary dataset $\mathtt{A}$, resulting in a kernel (covariance) whose weight-prior is tuned to suit f (as we assume a-priori that $\mathtt{A}$ and f share the same covariance structure).

Having derived a covariance $K_2^{\mathtt{A}}$ whose corresponding weight-space prior has been tuned to match the objective function f, we may proceed with standard Bayesian optimization modeling $f \sim \mathcal{GP}(0, K_2^{\mathtt{A}})$, where we posit that the more accurate matching of $K_2^{\mathtt{A}}$ to the covariance structure of f will accelerate the optimization process. In section 5 we demonstrate the efficacy of our algorithm in two applications, specifically (1) a test function with auxiliary data drawn from a flipped source and (2) new short polymer fibre design using micro-fluid devices (where auxiliary data is generated by an older, distinct device).

Our main contributions are:

- Definition of free-kernels and interpretation of free kernel weights as weight priors for Gaussian Processes.
- Weight-prior tuning using an auxiliary dataset.
- Accelerated Bayesian optimization using tuned weight priors (algorithm 1).
- Application of accelerated Bayesian optimization to real-world scenarios (section 5).

1.1 Notation

We use $\mathbb{N} = \{0, 1, \ldots\}$, $\bar{\mathbb{N}} = \mathbb{N} \cup \{\infty\}$. Column vectors are $\mathbf{a}, \mathbf{b}$, matrices $\mathbf{V}, \mathbf{W}$ (elements a_i, $W_{i,j}$). $\mathbf{a} \odot$

[1] And more generally reproducing kernel Banach space (RKBS) theory [Zhang et al., 2009, Fasshauer et al., 2015].

$\mathbf{b}$ is the element-wise product, $\mathbf{a}^{\odot b}$ the element-wise power, $|\mathbf{a}|$ the element-wise absolute, and $\mathrm{sum}(\mathbf{a}) = \sum_i a_i$. The m-dot-product $(\ldots)_m : (\mathbb{R}^n)^m \to \mathbb{R}$ is $(\mathbf{a}, \ldots, \mathbf{a}'''')_m = \mathrm{sum}(\mathbf{a} \odot \ldots \odot \mathbf{a}'''')$ [Dragomir, 2004, Salzo and Suykens, 2016], so $(\mathbf{x}, \mathbf{x}')_2$ is the dot product. The number of elements in a finite set $\mathtt{D}$ is denoted $|\mathtt{D}|$.

2 Problem Statement

In this paper we are concerned with solving the problem:

$$\mathbf{x} = \mathrm{argmax}_{\mathbf{x} \in \mathbb{X}} f(\mathbf{x})$$

where f is expensive to evaluate. We further suppose that we have been given an auxiliary set of training data $\mathtt{A} = \{(\mathbf{x}_i^{\mathtt{A}}, y_i^{\mathtt{A}}) \in \mathbb{X} \times \mathbb{R} | y_2^{\mathtt{A}} = h(g(\mathbf{x}_i^{\mathtt{A}}) + \nu_i), \nu_i \sim \mathcal{N}(0, \zeta^2)\}$ to accelerate the process, where h in this expression depends on the type of data represented by $\mathtt{A}$ (e.g. if $\mathtt{A}$ is binary classification data then $h(z) = \mathrm{sgn}(z)$, and if $\mathtt{A}$ is regression data then $h(z) = z$). The dataset $\mathtt{A}$ may include distilled knowledge from relevant patents (if f is the for example the yield of an experiment), observations from an older iteration of f (if f is a refinement of a manufacturing process) or human-generated samples (experimenter intuition).

Importantly, we do not assume $\mathtt{A}$ is generated by the function we wish to optimize (i.e. $f \neq g$ in general), but rather that $f \sim \mathcal{GP}(0, \tilde{K})$ is a draw from a (zero mean) Gaussian process with covariance $\tilde{K}$ and g is well modeled using kernel $\tilde{K}$; and that f and g depend on similar *features* in the RKHS $\mathcal{H}_{\tilde{K}}$ (equivalently (isomorphically) features in Gaussian Hilbert space [Janson, 1997] induced by $\tilde{K}$).

We use Bayesian Optimization (BO) here as f is presumed expensive to evaluate. Bayesian Optimization [Snoek et al., 2012] is a form of sequential model-based optimization (SMBO) that aims to to find $\mathbf{x}^* = \mathrm{argmax}_{\mathbf{x}} f(\mathbf{x})$ with the least number of evaluations for an expensive (to evaluate) function $f : \mathbb{X} \to \mathbb{R}$.

3 Background and Definitions

The algorithm we present in this paper combines Bayesian optimization (BO) with a variant of the kernel trick known as the m-kernel trick to achieve better tuning of the covariance (kernel) based on auxiliary data. As practitioners may not be familiar with the m-kernel trick, in this section we provide a quick review of the kernel trick and its m-kernel extension and their application to support vector machines (SVMs). We also define families of m-kernels (free kernels) that have a useful property that will play a central role later.

3.1 Kernels and the Kernel Trick

In machine learning the so-called "kernel trick" [Cortes and Vapnik, 1995, Schölkopf and Smola, 2001, Cristianini and Shawe-Taylor, 2005] is a ubiquitous way to converting any linear algorithm that may be expressed in terms of dot products in input space into a non-linear algorithm by simply replacing all dot products $\mathbf{x}^{\mathrm{T}}\mathbf{x}'$ with kernel evaluations $K(\mathbf{x}, \mathbf{x}')$, where $K : \mathbb{X} \times \mathbb{X} \to \mathbb{R}$ ($\mathbb{X} \neq \mathbb{R}^n$ in general) is a positive definite kernel (called Mercer kernels here to prevent later ambiguity). By Mercer's condition [Mercer, 1909], corresponding to K is an (implicit) feature map $\boldsymbol{\varphi} : \mathbb{X} \to \mathbb{R}^d$, $d \in \bar{\mathbb{N}}$, such that $K(\mathbf{x}, \mathbf{x}') = \boldsymbol{\varphi}(\mathbf{x})^{\mathrm{T}} \boldsymbol{\varphi}(\mathbf{x}')$, so the kernel trick applies an (implicit) transform $\mathbf{x} \to \boldsymbol{\varphi}(\mathbf{x})$ to all inputs.

In Gaussian Processes (GPs) and Bayesian theory more generally the covariance function K is analogous to the kernel function in machine learning, though it is often conceived differently and distinctions may arise [Kanagawa et al., 2018]. The performance of a kernel in a given context depends on how well matched it and its attendant hyper-parameters are to the problem at hand. This matching problem is known as kernel selection and hyper-parameter tuning, and typically relies on heuristics, intuition(s)/prior(s), and optimization techniques such as grid search or Bayesian optimization.

In practice, the kernel K is usually selected from a set of well-known kernels such as the linear, polynomial, squared-exponential (SE) or Matérn kernels - and potentially combinations thereof (eg. multi-kernel learning [Lanckriet et al., 2004, Bach et al., 2004]) - where any kernel parameters are simultaneously tuned to optimize some measure of fit such as log-likelihood or cross-validation error. We note that by taking this approach the search is effectively limited to the subset of possible feature maps embodied by the set of possible kernels chosen, which is by no means guaranteed to approach optimal feature map.

3.2 m-Kernels and Free Kernels

Less well known is the m-kernel trick, which allows conversion of any linear algorithm that may be expressed in terms of m-dot-products [Dragomir, 2004] $(\mathbf{x}, \mathbf{x}', \ldots, \mathbf{x}'''')_m = \mathrm{sum}(\mathbf{x} \odot \mathbf{x}' \odot \ldots \odot \mathbf{x}'''')$, $m \in \mathbb{N}$, in input space into a non-linear algorithm. Similar to the kernel trick, it works by replacing m-dot-products $(\mathbf{x}, \mathbf{x}', \ldots, \mathbf{x}'''')_m$ with m-kernel evaluations $K_m(\mathbf{x}, \mathbf{x}', \ldots, \mathbf{x}'''')$, where in this case $K_m : \mathbb{X}^m \to \mathbb{R}$ is an m-kernel (tensor kernel [Salzo et al., 2018]) - that is, $K_m : \mathbb{X}^m \to \mathbb{R}$, and there exists an (implicit) feature map $\boldsymbol{\varphi}_m : \mathbb{X} \to \mathbb{R}^d$, $d \in \bar{\mathbb{N}}$, such that:

$$K_m(\mathbf{x}, \ldots, \mathbf{x}'''') = (\boldsymbol{\varphi}_m(\mathbf{x}), \ldots, \boldsymbol{\varphi}_m(\mathbf{x}''''))_m \quad (1)$$

So, like the kernel trick, the m-kernel trick works by applying an (implicit) transform $\mathbf{x} \to \boldsymbol{\varphi}_m(\mathbf{x})$ to all inputs. The canonical example of the m-kernel trick is the p-norm SVM. For instructive purposes we have included an introduction in the supplementary material. Mercer kernels are 2-kernels.

From a Bayesian perspective an m-kernel K_m is analogous to a moment function [Der and Lee, 2007]. The formal equivalence of the L^m-moment classifier [Der and Lee, 2007] and the ℓ^p-SVM [Salzo et al., 2018] may be seen by inspection. Like a Mercer kernel, the per-

	$K_m(\mathbf{x},\ldots,\mathbf{x}'''')$
Linear:	$(\mathbf{x},\mathbf{x}',\ldots,\mathbf{x}'''')_m$
Polynomial:	$((\mathbf{x},\mathbf{x}',\ldots,\mathbf{x}'''')_m+\nu)^p$
Hyperbolic sine:	$\sinh(\nu(\mathbf{x},\ldots,\mathbf{x}'''')_m)$
Exponential:	$\exp(\nu(\mathbf{x},\mathbf{x}',\ldots,\mathbf{x}'''')_m)$
Log ratio:	$\prod_i \ln\left(\frac{1+x_i x_i'\ldots x_i''''}{1-x_i x_i'\ldots x_i''''}\right)$
SE:	$\exp(\frac{\nu}{2}(2(\mathbf{x},\ldots,\mathbf{x}'''')_m-\sum_{\cdots}\|\mathbf{x}^{\cdots}\|_2^2))$

Table 1: Examples of (free) m-kernels ($\nu\geq 0$, $p\in\mathbb{N}$).

formance of an m-kernel depends on its hyper-parameters.

For the purposes of the present paper we define the following familes of m-kernels for which the feature map is independent, in the specified sense, of m:

Definition 1 (Free kernel) *A* free kernel *is a family of functions $K_m:\mathbb{X}^m\to\mathbb{R}$ indexed by $m=2,4,\ldots$ for which there exists an* unweighted *feature map $\boldsymbol{\vartheta}:\mathbb{X}\to\mathbb{R}^d$, $d\in\bar{\mathbb{N}}$, and feature weights $\boldsymbol{\tau}\in\mathbb{R}^d$, both independent of m, so:*

$$K_m(\mathbf{x},\ldots,\mathbf{x}'''')=\left(\boldsymbol{\tau}^{\odot 2},\boldsymbol{\vartheta}(\mathbf{x}),\ldots,\boldsymbol{\vartheta}(\mathbf{x}'''')\right)_{m+1} \quad (2)$$

For fixed m a free kernel defines (is) an m-kernel:

$$K_m(\mathbf{x},\ldots,\mathbf{x}'''')=(\boldsymbol{\varphi}_m(\mathbf{x}),\ldots,\boldsymbol{\varphi}_m(\mathbf{x}''''))_m \quad (3)$$

with implied feature map $\boldsymbol{\varphi}_m(\mathbf{x})=\boldsymbol{\tau}^{\odot\frac{2}{m}}\odot\boldsymbol{\vartheta}(\mathbf{x})$.

Like Mercer kernels, standard m-kernels may be built - e.g. if $\mathbb{X}=\mathbb{R}^n$ and $k:\mathbb{R}\to\mathbb{R}$ is expandable as a Taylor series $k(\chi)=\sum_i\xi_i\chi^i$ with $\xi_i\geq 0$ then:

$$K_{\bullet m}(\mathbf{x},\mathbf{x}',\ldots,\mathbf{x}'''')=k((\mathbf{x},\mathbf{x}',\ldots,\mathbf{x}'''')_m) \quad (4)$$

is an m-dot-product kernel, and:

$$K_{\odot m}(\mathbf{x},\mathbf{x}',\ldots,\mathbf{x}'''')=\textstyle\prod_i k(x_i x_i'\ldots x_i'''') \quad (5)$$

the m-direct-product kernel [Salzo et al., 2018]. A sample of m-kernels is presented in table 1. It is not difficult to see that the m-dot-product and m-direct-product kernels are free kernels with unweighted feature map $\vartheta_{\mathbf{i}}(\mathbf{x})=x_0^{i_0}x_1^{i_1}\ldots x_{n-1}^{i_{n-1}}$ ($\mathbf{i}\in\mathbb{N}^n$ is a multi-index) and weights:

$$\begin{aligned}\tau_{\bullet\mathbf{i}}&=\sqrt{\tfrac{\mathrm{sum}(\mathbf{i})!}{i_0!i_1!\ldots i_{n-1}!}\xi_{\mathrm{sum}(\mathbf{i})}}\\ \tau_{\odot\mathbf{i}}&=\sqrt{\xi_{i_0}\xi_{i_1}\cdots\xi_{i_{n-1}}}\end{aligned}$$

respectively. It follows that all of the m-kernels in table 1 are free kernels, where we note that the SE m-kernel extension given in the table has unweighted feature map $\boldsymbol{\vartheta}(\mathbf{x})=\|\boldsymbol{\tau}_e\odot\boldsymbol{\varphi}_e(\mathbf{x})\|_2^{-1}\boldsymbol{\vartheta}_e(\mathbf{x})$ and feature weights $\boldsymbol{\tau}=\boldsymbol{\tau}_e$, where $\boldsymbol{\vartheta}_e$ and $\boldsymbol{\tau}_e$ are the unweighted feature map and feature weights of the exponential m-dot-product kernel.

3.3 Kernel Methods and Representor Theory

Support vector machines (SVM) (and kernel methods more generally) are a family of techniques based around the concepts of structural risk minimization and the kernel trick [Cortes and Vapnik, 1995, Steinwart and Christman, 2008]. At its most basic, if $\mathtt{A}=\{(\mathbf{x}_i^{\mathtt{A}},y_i^{\mathtt{A}})\in\mathbb{X}\times\mathbb{R}\}$ is a training set then the aim is to find an model:[2]

$$g_{\mathtt{A}}(\mathbf{x})=\mathbf{w}_{\mathtt{A}}^{\mathrm{T}}\boldsymbol{\varphi}(\mathbf{x}) \quad (6)$$

where $\boldsymbol{\varphi}:\mathbb{X}\to\mathbb{R}^d$ is implied by a Mercer kernel K and the weights $\mathbf{w}_{\mathtt{A}}\in\mathbb{R}^d$ solve:

$$\min_{\mathbf{w}_{\mathtt{A}}\in\mathbb{R}^d}R=r\left(\tfrac{1}{2}\|\mathbf{w}_{\mathtt{A}}\|_2^2\right)+\tfrac{1}{\lambda}\textstyle\sum_i E(y_i^{\mathtt{A}},g_{\mathtt{A}}(\mathbf{x}_i^{\mathtt{A}})) \quad (7)$$

where r is strictly monotonically increasing, E is an empirical risk function, and λ controls the trade-off between empirical risk minimization and regularization. By representor theory [Steinwart and Christman, 2008]:

$$\exists\boldsymbol{\alpha}^{\mathtt{A}}\in\mathbb{R}^{|\mathtt{A}|}\text{ s.t. }\mathbf{w}_{\mathtt{A}}=\textstyle\sum_i\alpha_i^{\mathtt{A}}\boldsymbol{\varphi}(\mathbf{x}_i^{\mathtt{A}}) \quad (8)$$

and hence $g_{\mathtt{A}}(\mathbf{x})=\sum_i\alpha_i^{\mathtt{A}}K(\mathbf{x},\mathbf{x}_i^{\mathtt{A}})$. Note that the weight vector $\mathbf{w}_{\mathtt{A}}$ is not (explicitly) present in this expression, and may be likewise removed from the training problem - for example in ridge regression (LS-SVM [Suykens et al., 2002]) we let $E(y,g)=\frac{1}{2}(y-g)^2$, $r(v)=v$, so (7) becomes:

$$\min_{\boldsymbol{\alpha}^{\mathtt{A}}\in\mathbb{R}^{|\mathtt{A}|}}\tfrac{1}{2}\boldsymbol{\alpha}^{\mathtt{A}\mathrm{T}}(\mathbf{K}_{\mathtt{A}}+\lambda\mathbf{I})\boldsymbol{\alpha}^{\mathtt{A}}-\mathbf{y}^{\mathtt{A}\mathrm{T}}\boldsymbol{\alpha}^{\mathtt{A}} \quad (9)$$

where $\mathbf{K}_{\mathtt{A}}\in\mathbb{R}^{|\mathtt{A}|\times|\mathtt{A}|}$, $K_{\mathtt{A}i,j}=K(\mathbf{x}_i^{\mathtt{A}},\mathbf{x}_j^{\mathtt{A}})$; and for binary classification $y_i^{\mathtt{A}}=\pm 1$ and we may choose $E(y,g)=\max\{0,1-yg\}$ (hinge loss), $r(v)=v$, so (7) becomes:

$$\min_{\boldsymbol{\alpha}^{\mathtt{A}}\in\mathbb{R}^{|\mathtt{A}|},\mathbf{0}\leq\mathbf{y}^{\mathtt{A}}\odot\boldsymbol{\alpha}^{\mathtt{A}}\leq\frac{1}{\lambda}\mathbf{1}}\tfrac{1}{2}\boldsymbol{\alpha}^{\mathtt{A}\mathrm{T}}\mathbf{K}_{\mathtt{A}}\boldsymbol{\alpha}^{\mathtt{A}}-\mathbf{1}^{\mathrm{T}}|\boldsymbol{\alpha}^{\mathtt{A}}| \quad (10)$$

where $|\boldsymbol{\alpha}^{\mathtt{A}}|\in\mathbb{R}^{|\mathtt{A}|}$ is the element-wise absolute of $\boldsymbol{\alpha}^{\mathtt{A}}$.

4 Gaussian Processes and Weight Prior Tuning with Free Kernels

A gaussian process is a distribution on a space of functions [MacKay, 1998, Rasmussen and Williams, 2006]. The following introduction takes the weight-space perspective [Rasmussen and Williams, 2006], but rather than modeling f as a linear function in the feature space defined by the feature map $\boldsymbol{\varphi}:\mathbb{X}\to\mathbb{R}$ implied by a given Mercer kernel $K:\mathbb{X}\times\mathbb{X}\to\mathbb{R}$, we instead model f as a linear function in the *unweighted* feature space defined by the *unweighted* feature map $\boldsymbol{\vartheta}:\mathbb{X}\to\mathbb{R}$ implied by a given *free kernel* (see definition 1) $K_m:\mathbb{X}^m\to\mathbb{R}$, with a weight prior $\mathbf{v}\sim\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma}_{\boldsymbol{\tau}})$ defined by the feature weights implied by K_m. As we show, this is identical to the usual derivation from a function-space perspective, but in weight space we obtain the key insight that the feature weights $\boldsymbol{\tau}\in\mathbb{R}^d$ implied by the free kernel K_m control the relative importance

[2] A bias term is often included here, so $g_{\mathtt{A}}(\mathbf{x})=\mathbf{w}_{\mathtt{A}}^{\mathrm{T}}\boldsymbol{\varphi}(\mathbf{x})+b$. Typically this results in an additional constraint on the dual (e.g. $\mathrm{sum}(\boldsymbol{\alpha}^{\mathtt{A}})=0$ for the examples given). Alternatively the bias may always be incorporated into K if required - precisely $\boldsymbol{\varphi}(\mathbf{x})\to[\boldsymbol{\varphi}(\mathbf{x});1]$, so the bias b is included in $\mathbf{w}_{\mathtt{A}}\to[\mathbf{w}_{\mathtt{A}};b]$ and the kernel is adjusted as $K(\mathbf{x},\mathbf{x}')\to K(\mathbf{x},\mathbf{x}')+1$.

of the different features. Moreover, as we will show subsequently, it is straightforward to generate, in a principled manner, free kernels with the same unweighted feature map but distinct feature weights, allowing us to *tune* the weight-prior of our Gaussian process to better model f and hence accelerate our Bayesian optimizer.

Let $K_m : \mathbb{X}^m \to \mathbb{R}$ be a (given) free kernel with implied (unweighted) feature map $\boldsymbol{\vartheta} : \mathbb{X} \to \mathbb{R}^d$ and feature weights $\boldsymbol{\tau} \in \mathbb{R}^d$. We model f using the unweighted feature map:

$$f(\mathbf{x}) = \mathbf{v}^{\mathrm{T}} \boldsymbol{\vartheta}(\mathbf{x}) \tag{11}$$

assuming a weight prior $\mathbf{v} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}_{\boldsymbol{\tau}})$, where $\boldsymbol{\Sigma}_{\boldsymbol{\tau}} = \mathrm{diag}(\boldsymbol{\tau}^{\odot 2})$. Following the usual method[3] we see that the posterior of $\mathbf{v}$ given observations $\mathtt{D} = \{(\mathbf{x}_i, y_i) | y_i = f(\mathbf{x}_i) + \epsilon_i, \epsilon_i \sim \mathcal{N}(0, \sigma^2)\}$ is $\mathbf{v}|\mathtt{D} \sim \mathcal{N}(\mathbf{m}_{\mathbf{v}|\mathtt{D}}, \boldsymbol{\Sigma}_{\mathbf{v}|\mathtt{D}})$:

$$\begin{aligned} \mathbf{m}_{\mathbf{v}|\mathtt{D}} &= \boldsymbol{\mu} + \boldsymbol{\Sigma}_{\boldsymbol{\tau}} \boldsymbol{\Theta}_{\mathtt{D}} \left(\boldsymbol{\Theta}_{\mathtt{D}}^{\mathrm{T}} \boldsymbol{\Sigma}_{\boldsymbol{\tau}} \boldsymbol{\Theta}_{\mathtt{D}} + \sigma^2 \mathbf{I} \right)^{-1} \mathbf{y} \\ \boldsymbol{\Sigma}_{\mathbf{v}|\mathtt{D}} &= \boldsymbol{\Sigma}_{\boldsymbol{\tau}} - \boldsymbol{\Sigma}_{\boldsymbol{\tau}} \boldsymbol{\Theta}_{\mathtt{D}} \left(\boldsymbol{\Theta}_{\mathtt{D}}^{\mathrm{T}} \boldsymbol{\Sigma}_{\boldsymbol{\tau}} \boldsymbol{\Theta}_{\mathtt{D}} + \sigma^2 \mathbf{I} \right)^{-1} \boldsymbol{\Theta}_{\mathtt{D}}^{\mathrm{T}} \boldsymbol{\Sigma}_{\boldsymbol{\tau}} \end{aligned} \tag{12}$$

where $\Theta_{\mathtt{D}i,j} = \vartheta_i(\mathbf{x}_j)$. Substituting (12) into (11) we find $f(\mathbf{x})|\mathtt{D} \sim \mathcal{N}(m_{f|\mathtt{D}}(\mathbf{x}), \Sigma_{f|\mathtt{D}}(\mathbf{x}, \mathbf{x}'))$:

$$\begin{aligned} m_{f|\mathtt{D}}(\mathbf{x}) &= \mu(\mathbf{x}) + \mathbf{k}_{\mathtt{D}}^{\mathrm{T}}(\mathbf{x}) \left(\mathbf{K}_{\mathtt{D}} + \sigma^2 \mathbf{I} \right)^{-1} \mathbf{y} \\ \Sigma_{f|\mathtt{D}}(\mathbf{x}, \mathbf{x}') &= K(\mathbf{x}, \mathbf{x}') - \mathbf{k}_{\mathtt{D}}^{\mathrm{T}}(\mathbf{x}) \left(\mathbf{K}_{\mathtt{D}} + \sigma^2 \mathbf{I} \right)^{-1} \mathbf{k}_{\mathtt{D}}(\mathbf{x}') \end{aligned} \tag{13}$$

where $\mathbf{y}, \mathbf{k}_{\mathtt{D}}(\mathbf{x}) \in \mathbb{R}^{|\mathtt{D}|}$, $\mathbf{K}_{\mathtt{D}} \in \mathbb{R}^{|\mathtt{D}| \times |\mathtt{D}|}$, $\mu(\mathbf{x}) := \boldsymbol{\mu}^{\mathrm{T}} \boldsymbol{\vartheta}(\mathbf{x})$, $k_{\mathtt{D}i}(\mathbf{x}) = K_2(\mathbf{x}, \mathbf{x}_i)$, and $K_{\mathtt{D}i,j} = K_2(\mathbf{x}_i, \mathbf{x}_j)$.

It is worth noting that all of the m-dot product and m-direct-product free kernels share the same unweighted feature map. When used in the above, then, all m-dot-product and m-direct-product free kernels model f as a linear function in the same (unweighted) feature space but apply different weight priors $\mathbf{v} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}_{\boldsymbol{\tau}})$. Moreover as the following theorem demonstrates, it is possible to *overwrite* the feature weights $\boldsymbol{\tau}$ for a given free-kernel, and hence apply arbitrary (diagonal) weight-priors on our GP.

Theorem 1 *Let K_m be a free kernel with unweighted feature map $\boldsymbol{\vartheta}$ and feature weights $\boldsymbol{\tau}$, and let $\mathtt{A}^* = \{(\mathbf{x}_i^{\mathtt{A}}, \alpha_i^{\mathtt{A}}) \in \mathbb{X} \times \mathbb{R}\}$. Then the family of functions:*

$$K_m^{\mathtt{A}}(\mathbf{x}, \mathbf{x}', \ldots) = \sum_{i,j} \alpha_i^{\mathtt{A}} \alpha_j^{\mathtt{A}} K_{m+2}\left(\mathbf{x}_i^{\mathtt{A}}, \mathbf{x}_j^{\mathtt{A}}, \mathbf{x}, \mathbf{x}', \ldots\right) \tag{14}$$

indexed by m are a free kernel with unweighted feature map $\boldsymbol{\vartheta}_{\mathtt{A}} = \boldsymbol{\vartheta}$, and feature weights $\boldsymbol{\tau}_{\mathtt{A}} = \boldsymbol{\tau} \odot \sum_i \alpha_i^{\mathtt{A}} \boldsymbol{\vartheta}(\mathbf{x}_i^{\mathtt{A}})$.

Proof: Using definition 1:

$$\begin{aligned} &K_m^{\mathtt{A}}(\mathbf{x}, \mathbf{x}', \ldots) = \textstyle\sum_{i,j} \alpha_i^{\mathtt{A}} \alpha_j^{\mathtt{A}} K_{m+2}(\mathbf{x}_i^{\mathtt{A}}, \mathbf{x}_j^{\mathtt{A}}, \mathbf{x}, \mathbf{x}', \ldots) \\ &= \textstyle\sum_{i,j} \alpha_i^{\mathtt{A}} \alpha_j^{\mathtt{A}} \left(\boldsymbol{\tau}^{\odot 2}, \boldsymbol{\vartheta}(\mathbf{x}_i^{\mathtt{A}}), \boldsymbol{\vartheta}(\mathbf{x}_j^{\mathtt{A}}), \boldsymbol{\vartheta}(\mathbf{x}), \boldsymbol{\vartheta}(\mathbf{x}'), \ldots\right)_{m+3} \\ &= \left((\boldsymbol{\tau} \odot \textstyle\sum_i \alpha_i^{\mathtt{A}} \boldsymbol{\vartheta}(\mathbf{x}_i^{\mathtt{A}}))^{\odot 2}, \boldsymbol{\vartheta}(\mathbf{x}), \boldsymbol{\vartheta}(\mathbf{x}'), \ldots \right)_{m+1} \\ &= \left(\boldsymbol{\tau}_{\mathtt{A}}^{\odot 2}, \boldsymbol{\vartheta}_{\mathtt{A}}(\mathbf{x}), \boldsymbol{\vartheta}_{\mathtt{A}}(\mathbf{x}'), \ldots \right)_{m+1} \end{aligned}$$ □

[3] For example following [Rasmussen and Williams, 2006], the differences in the derivation are entirely cosmetic at this point, with $\boldsymbol{\vartheta}$ replacing $\boldsymbol{\varphi}$, $\mathbf{v}$ replacing $\mathbf{w}$, and $\boldsymbol{\Sigma}_{\boldsymbol{\tau}}$ replacing $\boldsymbol{\Sigma}_p$.

The implication of this is that, starting with a free kernel K_m, we may train an SVM on our auxiliary dataset $\mathtt{A}$ and K_2 to obtain a model (in terms of unweighted feature map):

$$g_{\mathtt{A}}(\mathbf{x}) = \mathbf{v}_{\mathtt{A}}^{\mathrm{T}} \boldsymbol{\vartheta}(\mathbf{x})$$

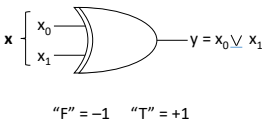
which by representor theory is parameterized by $\boldsymbol{\alpha}^{\mathtt{A}} \in \mathbb{R}^{|\mathtt{A}|}$:

$$\mathbf{v}_{\mathtt{A}} = \boldsymbol{\tau} \odot \textstyle\sum_i \alpha_i^{\mathtt{A}} \boldsymbol{\vartheta}(\mathbf{x}_i^{\mathtt{A}}) \tag{15}$$

Each element $v_{\mathtt{A}i}$ of $\mathbf{v}_{\mathtt{A}}$ controls the relative influence of that feature on the model - that is, a belief regarding the relative importance of feature i, where a large value $v_{\mathtt{A}i}$ implies and important feature - and theorem 1 provides a mechanism to directly build this belief (prior) into a Gaussian process via the construction of Mercer kernel $K_2^{\mathtt{A}}$. The following example illustrates the operation:

Example 1 (XOR features) Let the auxiliary dataset $\mathtt{A}$ be the famous XOR training set given by:

i	$\mathbf{x}_i^{\mathtt{A}}$	$y_i^{\mathtt{A}}$
0	$[-1, -1]$	-1
1	$[+1, -1]$	$+1$
2	$[-1, +1]$	$+1$
3	$[+1, +1]$	-1



Let $K_m(\mathbf{x}, \mathbf{x}', \ldots) = ((\mathbf{x}, \mathbf{x}', \ldots)_m + 1)^2$ with $\mathbf{x}, \mathbf{x}', \ldots \in \mathbb{R}^2$ be the quadratic m-kernel, which may be readily seen to have an unweighted feature map $\boldsymbol{\vartheta}(\mathbf{x}) = [1, x_0, x_1, x_0^2, x_1^2, x_0 x_1]$ and feature weights $\boldsymbol{\tau} = [1, \sqrt{2}, \sqrt{2}, 1, 1, \sqrt{2}]$. Note that the only feature relevant to the XOR auxiliary training set is $x_0 x_1$, as $x_0 \veebar x_1 = x_0 x_1$ (recall $+1$ is "T" and -1 is "F" in this representation).

If we train the SVM classifier (10) with quadratic kernel K_2 and $C = 1$ on $\mathtt{A}$ we find $\boldsymbol{\alpha}^{\mathtt{A}} = [-\frac{1}{8}, \frac{1}{8}, \frac{1}{8}, -\frac{1}{8}]$, and hence it is readily shown that the trained model (6) reduces to $g_{\mathtt{A}}(\mathbf{x}) = x_0 x_1$. By theorem 1 (in particular (14)) we may use $\mathtt{A}$ and $\boldsymbol{\alpha}^{\mathtt{A}}$ to construct the (modified) kernel:

$$\begin{aligned} &K_2^{\mathtt{A}}(\mathbf{x}, \mathbf{x}') = \textstyle\sum_{i,j=0,1,2,3} \alpha_i^{\mathtt{A}} \alpha_j^{\mathtt{A}} K_4\left(\mathbf{x}_i^{\mathtt{A}}, \mathbf{x}_j^{\mathtt{A}}, \mathbf{x}, \mathbf{x}'\right) \\ &= \tfrac{1}{64} \textstyle\sum_{i,j=0,1,2,3} \left(1 + \mathbf{x}^{\mathrm{T}} \left(\mathbf{x}_i^{\mathtt{A}} \mathbf{x}_j^{\mathtt{A}\mathrm{T}}\right) \mathbf{x}'\right)^2 = \tfrac{1}{2} x_0 x_1 x_0' x_1' \end{aligned}$$

This constructed kernel has the same unweighted feature map $\boldsymbol{\vartheta}_{\mathtt{A}}(\mathbf{x}) = [1, x_0, x_1, x_0^2, x_1^2, x_0 x_1]$ as the quadratic m-kernel K_m from which it was derived, but with feature weights $\boldsymbol{\tau}_{\mathtt{A}} = [0, 0, 0, 0, 0, \frac{1}{\sqrt{2}}]$ (as per theorem 14). This is reasonable in this case: as noted previously, $x_0 x_1$ is the only relevant feature for the XOR training set, and the only feature with non-zero weight in the constructed kernel $K_2^{\mathtt{A}}$.

Used as a covariance for a GP, $K_2^{\mathtt{A}}$ applies a weight prior $\mathbf{v} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\boldsymbol{\tau}_{\mathtt{A}}})$, $\boldsymbol{\Sigma}_{\boldsymbol{\tau}_{\mathtt{A}}} = \mathrm{diag}([0, 0, 0, 0, 0, \frac{1}{\sqrt{2}}])$, which asserts a belief (derived from the auxiliary dataset $\mathtt{A}$) that only the weight corresponding to feature $x_0 x_1$ is important.[4]

[4] In most real-world examples we may expect that the feature weights will not be so sparse (as the SVM applies 2-norm regularization on its weights rather than Lasso), and hence the belief asserted on the GP will be less dogmatic.

5 Accelerating Bayesian Optimization

Recall that our aim is to solve the problem:

$$\mathbf{x} = \text{argmax}_{\mathbf{x}\in\mathbb{X}} f(\mathbf{x})$$

where f is expensive to evaluate and we are given an auxiliary dataset $\mathtt{A} = \{(\mathbf{x}_i^{\mathtt{A}}, y_i^{\mathtt{A}}) \in \mathbb{X}\times\mathbb{R} | y_i^{\mathtt{A}} = h(g(\mathbf{x}_i^{\mathtt{A}}) + \nu_i), \nu_i \sim \mathcal{N}(0,\zeta^2)\}$ to accelerate the process (where $\mathtt{A}$ may include distilled knowledge e.g. from relevant patents, observations from related scenarios, human intuitions etc). As noted previously, we assume that $f \sim \mathcal{GP}(0,\tilde{K})$ is a draw from a (zero mean) Gaussian process and that g is well modeled using kernel $\tilde{K}$; and that f and g depend on similar *features* in the RKHS $\mathcal{H}_{\tilde{K}}$ induced by $\tilde{K}$.

As discussed in section 4, for a free kernel K_m we may extract weight priors from the auxiliary training set $\mathtt{A}$ by training an SVM, and then insert this as a prior in a Gaussian Process by constructing a kernel $K_2^{\mathtt{A}}$ using theorem 1. Taking as given that our assumptions regarding f and g are correct, the model $f \sim \mathcal{GP}(0, K_2^{\mathtt{A}})$ will have more appropriate priors than a model $f \sim \mathcal{GP}(0, K)$ for a "standard" kernel K, as the weight prior reflects insight gained into the relative importance of the different features, and thus better modeling of f using this derived kernel should improve the convergence rate of the Bayesian optimization of f.

Our algorithm is presented in algorithm 1. As noted previously, algorithm 1 does not assume direct knowledge of the covariance K of the Gaussian process from which f is drawn - rather, it infers (learns) K from the auxiliary dataset $\mathtt{A}$ before proceeding. Unlike DT-MKL [Duan et al., 2012], this search is not limited to the space spanned by a basis set of standard kernels - rather it is the space spanned by the unweighted feature map $\boldsymbol{\vartheta}$ of the m-kernel K_m. For example if we use an exponential m-kernel then the search space is the space of all dot-product Mercer kernels.

Algorithm 1 Tuned-Prior Bayesian Optimization.

input Auxiliary dataset $\mathtt{A}$, initial observations $\mathtt{D}_0$ of f.
input Free kernel K_m (for examples see table 1)
Train SVM with dataset $\mathtt{A}$, kernel K_2 to obtain $\boldsymbol{\alpha}^{\mathtt{A}} \in \mathbb{R}^{|\mathtt{A}|}$.
Construct re-weighted kernel $K_2^{\mathtt{A}}$:

$$K_2^{\mathtt{A}}(\mathbf{x},\mathbf{x}') = \textstyle\sum_{i,j} \alpha_i^{\mathtt{A}}\alpha_j^{\mathtt{A}} K_4(\mathbf{x},\mathbf{x}',\mathbf{x}_i^{\mathtt{A}},\mathbf{x}_j^{\mathtt{A}})$$

Modeling $f \sim \mathcal{GP}(0, K_2^{\mathtt{A}})$, proceed:
for $t = 0, 1, \ldots, T-1$ **do**
  Select test point $\mathbf{x}_t = \text{argmax}_{\mathbf{x}} a_t(\mathbf{x})$ (see (16)).
  Perform Experiment $y_t = f(\mathbf{x}_t) + \epsilon$, $\epsilon \sim \mathcal{N}(0,\nu)$.
  Update $\mathtt{D}_{t+1} := \mathtt{D}_t \cup \{(\mathbf{x}_t, y_t)\}$.
end for

For the acquisition function a_t we tested expected improvement (EI) [Mockus, 2002, Jones et al., 1998] and GP upper confidence bound (GP-UCB) [Srinivas et al., 2012]:

$$\begin{aligned} a_t^{\text{EI}} &= (m_{f|\mathtt{D}_t}(\mathbf{x}) - y^+)\Phi(Z_t(\mathbf{x})) + \sigma_{f|\mathtt{D}_t}(\mathbf{x})\phi(Z_t(\mathbf{x})) \\ a_t^{\text{UCB}} &= m_{f|\mathtt{D}_t}(\mathbf{x}) + \sqrt{\beta_t}\sigma_{f|\mathtt{D}_t}(\mathbf{x}) \end{aligned} \quad (16)$$

where $Z_t(\mathbf{x}) = (m_{f|\mathtt{D}_t}(\mathbf{x}) - y^+)/\sigma_{f|\mathtt{D}_t}(\mathbf{x})$; $\sigma^2_{f|\mathtt{D}_t}(\mathbf{x}) = \Sigma_{f|\mathtt{D}_t}(\mathbf{x},\mathbf{x})$; $\phi(\cdot)$ and $\Phi(\cdot)$ are the PDF and CDF functions for the normal distribution; $y^+ = \max_i\{y_i\}$; and β_t are constants [Srinivas et al., 2012]. We denote the variants of our algorithm using these acquisition functions, respectively, as TP-EI-BO and TP-GPUCB-BO.

Note that algorithm 1 is divided into two distinct steps, specifically pre-training using the auxiliary dataset $\mathtt{A}$ to obtain the covariance $K_2^{\mathtt{A}}$, and Bayesian Optimization using this covariance to model $f \sim \mathcal{GP}(0, K_2^{\mathtt{A}})$. We have used standard BO here, but in general any variant of BO using this model could be substituted as required.

6 Experimental Results

We now present a number of experiments applying algorithm 1 using GP-UCB and EI acquisition functions (TP-GPUCB-BO and TP-EI-BO), using a normalized SE free kernel with hyper-parameters (kernel length-scale and regularisation parameter λ) selected to minimize leave-one-out (LOO) error on the auxiliary dataset $\mathtt{A}$ during pre-training. We have compared our algorithm with Bayesian optimization using a standard squared-exponential covariance function (GPUCB and EI)[5], as well as standard transfer learning (envGPUCB and diffGPUCB [Joy et al., 2016, Shilton et al., 2017][6]), Bayesian optimization using a covariance function learned via Domain-Transfer Multi-Kernel Learning (DT-MKL-GPUCB and DT-MKL-EI) [Duan et al., 2012] using the kernel mixture:

$$K(\mathbf{x},\mathbf{x}') = \textstyle\sum_{i=0,1,2} v_i K_i(\mathbf{x},\mathbf{x}') \quad (17)$$

where K_0 is an SE kernel, K_1 a Matérn 1/2 kernel, K_2 a Matérn 3/2 kernel (all hyper-parameters in (17) were selected to minimize LOO error on the auxiliary dataset $\mathtt{A}$ in pre-training), and Bayesian optimization using an ARD-SE kernel with hyperparameters tuned to minimize LOO error on the auxilliary dataset (ARD-GPUCB and ARD-EI). All experiments were normalized to $f, h\circ g : [-1,1]^n \to [0,1]$.

All experiments run with SVMHeavy v7 [Shilton, 2020] (code available at https://github.com/apshsh/SVMHeavy).

6.1 Experiment 1: Flipped Test Function

In this experiment we consider the optimization of the (2-dimensional) Hölder-Table, Himmelblau, Ackley, Styblinski-Tang, eggholder and Rastringin test functions. In these experiments 50 auxiliary datapoints were drawn from the flipped (negated) test function $h(g(\mathbf{x})) = -f(\mathbf{x})$ to make $\mathtt{A}$. These were chosen for their non-trivial structure.

Results are shown in figure 1. As expected in this situation, standard transfer learning algorithms (envGPUCB, diffGPUCB, envEI and diffEI) typically (though curiously not

[5] For GPUCB and EI the hyper-parameters were tuned to minimize LOO error on $\mathtt{D}_t$ at each iteration t.

[6] We also include envEI and diffEI, which are like envGPUCB and diffGPUCB but using the EI acquisition function.

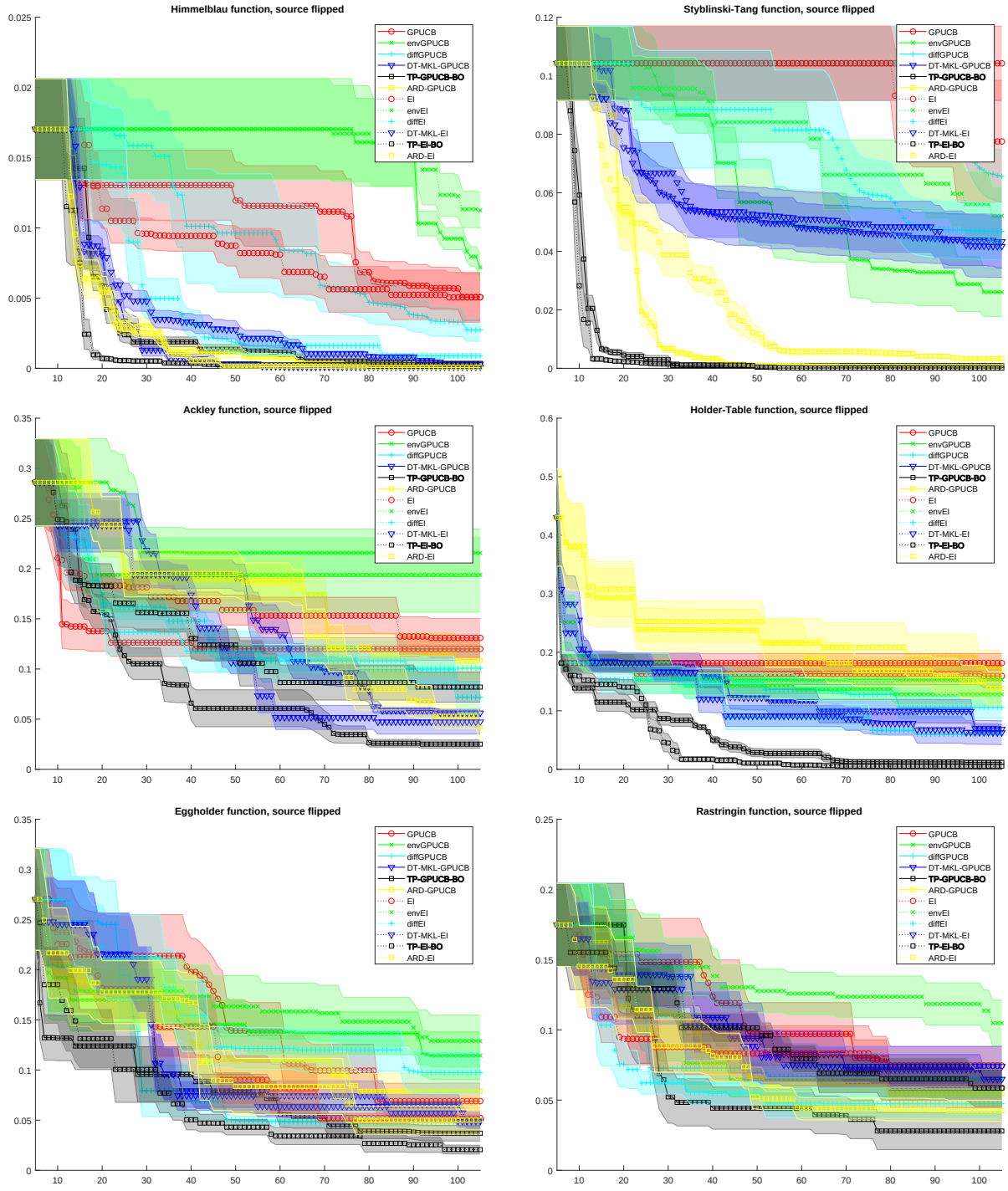


Figure 1: Convergence for test functions with flipped auxiliary data $f = -h \circ g$. Variants of our method are shown in black.

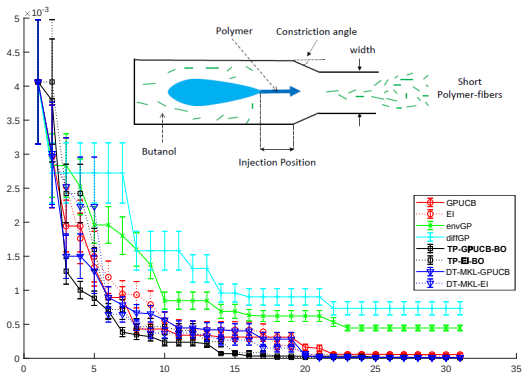


Figure 2: Short Polymer Fibre design. Comparison of algorithms in terms of minimum squared distance from the set target versus iterations. Device geometry for short polymer fibre injection shown as inset.

universally) perform worse than standard Bayesian optimization due to misdirecting auxiliary data; whereas DT-MKL helps convergence for all but the Rastringin function. Our method performs well in all cases shown.

6.2 Experiment 2: Short Polymer Fibres

In this experiment we have tested our algorithm on the real-world application of optimizing short polymer fibre (SPF) to achieve a given (median) target length [Li et al., 2017]. This process involves the injection of one polymer into another in a special device [Sutti et al., 2014]. The process is controlled by 3 geometric parameters (channel width (mm), constriction angle (degree), device position (mm)) and 2 flow factors (butanol speed (ml/hr), polymer concentration (cm/s)) that parametrize the experiment (figure 2). Two devices (A and B) were used. Device A is armed by a gear pump and allows for three butanol speeds (86.42, 67.90 and 43.21), device B has a lobe pump and allows butonal speed 98, 63 and 48. Our goal is to design a new short polymer fibre using Device B with median fibre length 500μm, so:

- Target: we aim to minimize $f(\mathbf{x}) = (g_B(\mathbf{x}) - 500)^2$, where $g_B(\mathbf{x})$ is the median fibre length (μm) produced by device B with settings $\mathbf{x} \in \mathbb{R}^5$ as described above. For simulation purposes this is represented by a grid of 155 experimental observations from [Li et al., 2017].
- Auxiliary dataset: $\mathtt{A}$ consists of 162 experimental observations of device A of the form $y_i^{\mathtt{A}} = (g_A(\mathbf{x}_i^{\mathtt{A}}) - 500)^2$, where $g_A(\mathbf{x})$ is the median fibre length produced by device A with $\mathbf{x} \in \mathbb{R}^5$ [Li et al., 2017].

Results are shown in figure 2. We note that, for this experiment, neither diffGP or envGP perform well, presumably due to the dissimilarities between g_A and g_B (but not their covariance structure). We note that our algorithm converges more quickly than the alternatives.

7 Notes and Future Directions

In the experiments we found that our method works well provided that the auxiliary data is sufficient to characterize the covariance of the optimization problem. As counter-examples we found that test functions that vary significantly on small scales (eg. the Levi N.13 test function), and test functions with sparse, sharp features that are easily missed (eg. the Easom test function), our method did not improve convergence (see supplementary). In the first case (Levi N.13) this is due to the inability to characterize such short-scale variation from the (comparatively) small dataset $\mathtt{A}$, the remedy for which is increasing the size of $\mathtt{A}$ (but see shortly). In the second case (Easom) difficulties arise due to the fact that the target is mostly flat, so $\mathtt{A}$ is constant for all points and subsequently we find $\boldsymbol{\alpha}^{\mathtt{A}} = \mathbf{0}$ (and hence $K_2^{\mathtt{A}}(\mathbf{x}, \mathbf{x}') = 0$ everywhere). We note that the vanishing kernel is easily detected, and moreover that all methods considered failed to make headway on this objective.

With regard to practicality, the computational cost of evaluating $K_2^{\mathtt{A}}$ is $O(|\mathtt{A}|^2)$, so the cost of solving $\mathbf{x}_t = \operatorname{argmax}_{\mathbf{x}} a_t(\mathbf{x})$ is $O(|\mathtt{A}|^2|\mathtt{D}_T|^2 + |\mathtt{D}_T|^3 + n(|\mathtt{A}|^2|\mathtt{D}_T| + |\mathtt{D}_T|^2))$, where term 1 is the (one-off) cost of computing $\mathbf{K}_{\mathtt{D}}$, term 2 is the (one-off) cost of inverting same (this is cached), and term 3 is the cost of evaluating the GP n times for global optimization (calculating $\mathbf{k}_{\mathtt{D}}(\mathbf{x})$ and matrix multiplications). We found this acceptable provided that $|\mathtt{A}|$ is not too large ($|\mathtt{A}| \lesssim 200$ appears reasonable), but for larger auxiliary datasets some form of sparsification (e.g. [Demontis et al., 2016]) will presumably be required.

Finally it is desirable to investigate the impact of weight-prior tuning on the regret bounds associated with whichever acquisition function is selected. It is unclear at present how to proceed in this direction, so this remains an open problem.

8 Conclusion

In this paper we presented an algorithm that can use auxiliary data to construct a GP covariance corresponding to a more appropriate weight prior for the objective function than is present in "standard" covariance functions. Using this, we have shown how BO may be accelerated by modeling the objective function using this covariance function. We have demonstrated our algorithm on a practical applications (short polymer fibre manufacture) and a number of test functions. Finally we have discussed the applicability of our algorithm and future research directions.

Acknowledgments

This research was partially funded by the Australian Government through the Australian Research Council (ARC). Prof Venkatesh is the recipient of an ARC Australian Laureate Fellowship (FL170100006).

References

[Bach et al., 2004] Bach, F. R., Lanckriet, G. R. G., and Jordan, M. I. (2004). Multiple kernel learning, conic duality and the SMO algorithm. In *Proceedings of the 21st International Conference on Machine Learning (ICML)*, Banff.

[Bardenet et al., 2013] Bardenet, R., Brendel, M., Kégl, B., , and Sebag, M. (2013). Collaborative hyperparameter tuning. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 199–207.

[Brochu et al., 2010] Brochu, E., Cora, V. M., and de Freitas, N. (2010). A tutorial on bayesian optimization of expensive cost functions, with applications to active user modeling and heirarchical reinforcement learning. eprint arXiv:1012.2599, arXiv.org.

[Cortes and Vapnik, 1995] Cortes, C. and Vapnik, V. (1995). Support vector networks. *Machine Learning*, 20(3):273–297.

[Cristianini and Shawe-Taylor, 2005] Cristianini, N. and Shawe-Taylor, J. (2005). *An Introductino to Support Vector Machines and other Kernel-Based Learning Methods*. Cambridge University Press, Cambridge, UK.

[Demontis et al., 2016] Demontis, A., Melis, M., Biggio, B., Fumera, G., and Roli, F. (2016). Super-sparse learning in similarity spaces. *IEEE Computational Intelligence Magazine*, 11(4):36–45.

[Der and Lee, 2007] Der, R. and Lee, D. (2007). Large-margin classification in banach spaces. In *Proceedings of the JMLR Workshop and Conference 2: AISTATS2007*, pages 91–98.

[Dragomir, 2004] Dragomir, S. S. (2004). *Semi-inner products and Applications*. Hauppauge, New York.

[Duan et al., 2012] Duan, L., Tsang, I. W., and Xu, D. (2012). Domain transfer multiple kernel learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(3):465–479.

[Fasshauer et al., 2015] Fasshauer, G. E., Hickernell, F. J., and Ye, Q. (2015). Solving support vector machines in reproducing kernel banach spaces with positive definite functions. *Applied and Computational Harmonic Analysis*, 38(1):115–139.

[Janson, 1997] Janson, S. (1997). *Gaussian Hilbert Spaces*. Cambridge University Press.

[Jones et al., 1998] Jones, D. R., Schonlau, M., and Welch, W. J. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492.

[Joy et al., 2016] Joy, T. T., Rana, S., Gupta, S. K., and Venkatesh, S. (2016). Flexible transfer learning framework for bayesian optimisation. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 102–114. Springer.

[Kanagawa et al., 2018] Kanagawa, M., Hennig, P., Sejdinovic, D., and Sriperumbudur, B. K. (2018). Gaussian processes and kernel methods: A review on connections and equivalences. *arXiv preprint arXiv:1807.02582*.

[Lanckriet et al., 2004] Lanckriet, G. R. G., Cristianini, N., Bartlett, P., Ghaoui, L. E., and Jordan, M. I. (2004). Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72.

[Li et al., 2017] Li, C., de Celis Leal, D. R., Rana, S., Gupta, S., Sutti, A., Greenhill, S., Slezak, T., Height, M., and Venkatesh, S. (2017). Rapid bayesian optimisation for synthesis of short polymer fiber materials. *Scientific reports*, 7(1):5683.

[MacKay, 1998] MacKay, D. J. C. (1998). Introduction to gaussian processes. *NATO ASI Series F Computer and Systems Sciences*, 168.

[Mercer, 1909] Mercer, J. (1909). Functions of positive and negative type, and their connection with the theory of integral equations. *Transactions of the Royal Society of London*, 209(A).

[Micchelli et al., 2006] Micchelli, C. A., Xu, Y., and Zhang, H. (2006). Universal kernels. *Journal of Machine Learning Research*, 7.

[Mockus, 2002] Mockus, J. (2002). Bayesian heuristic approach to global optimization and examples. *Journal of Global Optimization*, 22(1–4):191–203.

[Pan and Yang, 2010] Pan, S. J. and Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359.

[Rasmussen, 2006] Rasmussen, C. E. (2006). Gaussian processes for machine learning. MIT Press.

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

[Salzo and Suykens, 2016] Salzo, S. and Suykens, J. A. K. (2016). Generalized support vector regression: duality and tensor-kernel representation. *arXiv preprint arXiv:1603.05876*.

[Salzo et al., 2018] Salzo, S., Suykens, J. A. K., and Rosasco, L. (2018). Solving ℓ^p-norm regularization with tensor kernels. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS) 2018*.

[Schölkopf and Smola, 2001] Schölkopf, B. and Smola, A. J. (2001). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, Massachusetts.

[Shilton, 2020] Shilton, A. (2001–2020). SVMHeavy: SVM, machine learning and optimisation libary. https://github.com/apshsh/SVMHeavy.

[Shilton et al., 2017] Shilton, A., Gupta, S., Rana, S., and Venkatesh, S. (2017). Regret bounds for transfer learning in bayesian optimisation. In Singh, A. and Zhu, J., editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 307–315, Fort Lauderdale, FL, USA. PMLR.

[Snoek et al., 2012] Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. In *Proceedings of NIPS 2012*, pages 2951–2959.

[Srinivas et al., 2012] Srinivas, N., Krause, A., Kakade, S. M., and Seeger, M. W. (2012). Information-theoretic regret bounds for gaussian process optimization in the bandit setting. *IEEE Transactions on Information Theory*, 58(5):3250–3265.

[Sriperumbudur et al., 2011] Sriperumbudur, B. K., Fukumizu, K., and Lanckrie, G. R. G. (2011). Universality, characteristic kernels and rkhs embedding of measures. *Journal of Machine Learning Research*, 12.

[Steinwart and Christman, 2008] Steinwart, I. and Christman, A. (2008). *Support Vector Machines*. Springer.

[Sutti et al., 2014] Sutti, A., Kirkland, M., Collins, P., and John George, R. (2014). An apparatus for producing nano-bodies: Patent WO 2014134668 A1.

[Suykens et al., 2002] Suykens, J. A. K., Van Gestel, T., De Brabanter, J., De Moor, B., and Vandewalle, J. (2002). *Least Squares Support Vector Machines*. World Scientific Publishing, New Jersey.

[Yogatama and Mann, 2014] Yogatama, D. and Mann, G. (2014). Efficient transfer learning method for automatic hyperparameter tuning. In *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics (AISTATS)*.

[Zhang et al., 2009] Zhang, H., Xu, Y., and Zhang, J. (2009). Reproducing kernel banach spaces for machine learning. *Journal of Machine Learning Research*, 10:2741–2775.