High Dimensional Bayesian Optimisation and Bandits via Additive Models

Kirthevasan Kandasamy  
Jeff Schneider  
Barnabás Póczos  
Carnegie Mellon University, Pittsburgh, PA, USA

Abstract
Bayesian Optimisation (BO) is a technique used in optimising a $D$-dimensional function which is typically expensive to evaluate. While there have been many successes for BO in low dimensions, scaling it to high dimensions has been notoriously difficult. Existing literature on the topic are under very restrictive settings. In this paper, we identify two key challenges in this endeavour. We tackle these challenges by assuming an additive structure for the function. This setting is substantially more expressive and contains a richer class of functions than previous work. We prove that, for additive functions the regret has only linear dependence on $D$ even though the function depends on all $D$ dimensions. We also demonstrate several other statistical and computational benefits in our framework. Via synthetic examples, a scientific simulation and a face detection problem we demonstrate that our method outperforms naive BO on additive functions and on several examples where the function is not additive.

1. Introduction
In many applications we are tasked with zeroth order optimisation of an expensive to evaluate function $f$ in $D$ dimensions. Some examples are hyper parameter tuning in expensive machine learning algorithms, experiment design, optimising control strategies in complex systems, and scientific simulation based studies. In such applications, $f$ is a blackbox which we can interact with only by querying for the value at a specific point. Related to optimisation is the bandits problem arising in applications such as online advertising and reinforcement learning. Here the objective is to maximise the cumulative sum of all queries. In either case, we need to find the optimum of $f$ using as few queries as possible by managing exploration and exploitation.

Bayesian Optimisation (Mockus & Mockus, 1991) refers to a suite of methods that tackle this problem by modeling $f$ as a Gaussian Process (GP). In such methods the challenge is two fold. At time step $t$, first estimate the unknown $f$ from the query value-pairs. Then use it to intelligently query at $x_t$ where the function is likely to be high. For this, we first we use the posterior GP to construct an acquisition function $\varphi_t$ which captures the value of the experiment. Then we maximise $\varphi_t$ to determine $x_t$.

Gaussian process bandits and Bayesian optimisation (GPB/BO) have been successfully applied in many applications such as tuning hyperparameters in learning algorithms (Snoek et al., 2012; Bergstra et al., 2011; Mahendran et al., 2012), robotics (Lizotte et al., 2007; Martinez-Cantin et al., 2007) and object tracking (Denil et al., 2012). However, all such successes have been in low (typically < 10) dimensions (Wang et al., 2013). Expensive high dimensional functions occur in several problems in fields such as computer vision (Yamins et al., 2013), antenna design (Hornby et al., 2006), computational astrophysics (Parkinson et al., 2006) and biology (Gonzalez et al., 2014). Scaling GPB/BO methods to high dimensions for practical problems has been challenging. Even current theoretical results suggest that GPB/BO is exponentially difficult in high dimensions without further assumptions (Srinivas et al., 2010; Bull, 2011). To our knowledge, the only approach to date has been to perform regular GPB/BO on a low dimensional subspace. This works only under strong assumptions.

We identify two key challenges in scaling GPB/BO to high dimensions. The first is the statistical challenge in estimating the function. Nonparametric regression is inherently difficult in high dimensions with known lower bounds depending exponentially in dimension (Györfi et al., 2002). The often exponential sample complexity for regression is invariably reflected in the regret bounds for GPB/BO. The second is the computational challenge in maximising $\varphi_t$. Commonly used global optimisation heuristics used to maximise $\varphi_t$ themselves require computation exponential in dimension. Any attempt to scale GPB/BO to high dimensions must effectively address these two concerns.

In this work, we embark on this challenge by treating \( f \) as an additive function of mutually exclusive lower dimensional components. **Our contributions** in this work are:

1. We present the **Add-GP-UCB** algorithm for optimisation and bandits of an additive function. An attractive property is that we use an acquisition function which is easy to optimise in high dimensions.

2. In our theoretical analysis we bound the regret for **Add-GP-UCB**. We show that it has only linear dependence on the dimension \( D \) when the \( f \) is additive.

3. Empirically we demonstrate that **Add-GP-UCB** outperforms naive BO on synthetic experiments, an astrophysical simulator and the Viola and Jones face detection problem. Furthermore **Add-GP-UCB** does well on several examples when the function is not additive.

A Matlab implementation of our methods is available online at [github.com/kirthevasank/add-gp-ucb](https://github.com/kirthevasank/add-gp-ucb).

2. Related Work

GPB/ BO methods follow a family of GP based active learning methods which select the next experiment based on the posterior (Osborne et al., 2012; Ma et al., 2015; Kandasamy et al., 2015). In the GPB/ BO setting, common acquisition functions include Expected improvement (Mockus, 1994), probability of improvement (Jones et al., 1998), Thompson sampling (Thompson, 1933) and upper confidence bound (Auer, 2003). Of particular interest to us, is the Gaussian process upper confidence bound (GP-UCB). It was first proposed and analysed in the noiseless setting by Srinivas et al. (2010) and extended to the noiseless case by de Freitas et al. (2012).

To our knowledge, most literature for GPB/ BO in high dimensions are in the setting where the function varies only along a very low dimensional subspace (Chen et al., 2012; Wang et al., 2013; Djolonga et al., 2013). In these works, the authors do not encounter either challenge as they perform GPB/ BO in either a random or carefully selected lower dimensional subspace. However, assuming that the problem is an easy (low dimensional) one hiding in a high dimensional space is often too restrictive. Indeed, our experimental results confirm that such methods perform poorly on real applications when the assumptions are not met. While our additive assumption is strong in its own right, it is considerably more expressive. It is more general than the setting in Chen et al. (2012). Even though it does not contain the settings in Djolonga et al. (2013); Wang et al. (2013), unlike them, we still allow the function to vary along the entire domain.

Using an additive structure is standard in high dimensional regression literature both in the GP framework and other-wise. Hastie & Tibshirani (1990); Ravikumar et al. (2009) treat the function as a sum of one dimensional components. Our additive framework is more general. Duvenaud et al. (2011) assume a sum of functions of all combinations of lower dimensional coordinates. These literature argue that using an additive model has several advantages even if \( f \) is not additive. It is a well understood notion in statistics that when we only have a few samples, using a simpler model to fit our data may give us a better trade off for estimation error against approximation error. This observation is crucial: in many applications for Bayesian optimisation we are forced to work in the low sample regime since calls to the blackbox are expensive. Though the additive assumption is biased for nonadditive functions, it enables us to do well with only a few samples. While we have developed theoretical results only for additive \( f \), empirically we show that our additive model outperforms naive GPB/ BO even when the underlying function is not additive.

Analyses of GPB/ BO methods focus on the query complexity of \( f \) which is the dominating cost in relevant applications. It is usually assumed that \( \varphi_t \) can be maximised to arbitrary precision at negligible cost. Common techniques to maximise \( \varphi_t \) include grid search, Monte Carlo and multistart methods (Brochu et al., 2010). In our work we use the Dividing Rectangles (DiRect) algorithm of Jones et al. (1993). While these methods are efficient in low dimensions they require exponential computation in high dimensions. It is widely acknowledged in the community that this is a critical bottleneck in scaling GPB/ BO to high dimensions (de Freitas, 2014). While we still work in the paradigm where evaluating \( f \) is expensive and characterise our theoretical results in terms of query complexity, we believe that assuming arbitrary computational power to optimise \( \varphi_t \) is too restrictive. For instance, in hyperparameter tuning the budget for determining the next experiment is dictated by the cost of the learning algorithm. In online advertising and robotic reinforcement learning we need to act in under a few seconds or real time.

In this manuscript, Section 3 formally details our problem and assumptions. We present **Add-GP-UCB** in Section 4 and our theoretical results in Section 4.3. All proofs are deferred to Appendix B. We summarize the regrets for **Add-GP-UCB** and **GP-UCB** in Table 1. The experiments section is in Appendix C. Section 5 presents a summary.

3. Problem Statement & Set up

We wish to maximise a function \( f : \mathcal{X} \to \mathbb{R} \) where \( \mathcal{X} \) is a rectangular region in \( \mathbb{R}^D \). We will assume w.l.o.g \( \mathcal{X} = [0, 1]^D \). \( f \) may be nonconvex and gradient information is not available. We can interact with \( f \) only by querying at some \( x \in \mathcal{X} \) and obtain a noisy observation \( y = f(x) + \epsilon \). Let an optimum point be \( x^* = \arg\max_{x \in \mathcal{X}} f(x) \).
Suppose at time $t$ we choose to query at $x_t$. Then we incur instantaneous regret $r_t = f(x_t) - f(x_t)$. In the bandit setting, we are interested in the cumulative regret $R_T = \sum_{t=1}^T r_t = \sum_{t=1}^T f(x_t) - f(x_t)$, and in the optimisation setting we are interested in the simple regret $S_T = \min_{t \leq T} r_t = f(x_T) - \max_x f(x)$. Since $S_T \leq \frac{1}{T} R_T$ any procedure with bounds on the cumulative regret is also a consistent procedure for optimisation. For any algorithm, a desirable property is to have no regret: $\lim_{T \to \infty} \frac{1}{T} R_T = 0$.

**Key structural assumption:** In order to make progress in high dimensions, we will assume that $f$ decomposes into the following additive form,

$$f(x) = f^{(1)}(x^{(1)}) + f^{(2)}(x^{(2)}) + \cdots + f^{(M)}(x^{(M)}).$$

Here each $x^{(j)} \in \mathcal{X}^{(j)} = [0, 1]^{d_j}$ are lower dimensional components. We will refer to the $\mathcal{X}^{(j)}$’s as “groups” and the grouping of different dimensions into these groups is very large and the group dimensionality is bounded: $d_j \leq d \ll D$. We have $D = dM \geq \sum_j d_j$. Parenthesised superscripts index the groups and a union over the groups. We will assume that the variables is a $d^{th}$ order kernel. A kernel which acts only on $d$ variables a $d^{th}$ order kernel. A kernel which acts on all the variables is a $D^{th}$ order kernel. Our kernel for $f$ is a sum of $M$ at most $d^{th}$ order kernels which, we will show, is statistically simpler than a $D^{th}$ order kernel.

We conclude this section by looking at some seemingly straightforward approaches to tackle the problem. The first natural question is of course why not directly run **GP-UCB** using the additive kernel? Since it is simpler than a $D^{th}$ order kernel we can expect statistical gains. While this is true, it still requires optimising $\varphi$ in $D$ dimensions to determine the next point which is expensive.

Alternatively, for an additive function, we could adopt a sequential approach where we use $1/M$ fraction of our query budget to maximise the first group by keeping the rest of the coordinates constant. Then we proceed to the second group and so on. While optimising a $d$ dimensional acquisition function is easy, this approach is not desirable for several reasons. First, it will not be an anytime algorithm as we will have to pre-allocate our query budget to maximise each group. Once we proceed to a new group we cannot come back and optimise an older one. Second, such an approach places too much faith in the additive assumption. We will only have explored $M$ $d$-dimensional hyperplanes in the entire space. Third, it is not suitable as a bandit algorithm as we suffer high regret until we get to the last group. We will further elaborate on the deficiencies of this and other sequential approaches in Appendix A.2.
We have illustrated this graphically in Figure 1. The joint distribution of \( f(x) \) is itself a difficult problem. Fortunately, in our setting we can be more efficient. We propose an alternative acquisition function which applies to an additive kernel. We define the Additive Gaussian Process Upper Confidence Bound (Add-GP-UCB) to be

\[
\tilde{\varphi}_t(x) = \mu_{t-1}(x) + \beta_t^{1/2} \sigma_{t-1}(x^{(j)}).
\]  

We immediately see that we can write \( \tilde{\varphi}_t \) as a sum of functions on orthogonal domains: \( \tilde{\varphi}_t(x) = \sum_j \tilde{\varphi}_t^{(j)}(x^{(j)}) \) where \( \tilde{\varphi}_t^{(j)}(x^{(j)}) = \mu_{t-1}(x^{(j)}) + \beta_t^{1/2} \sigma_{t-1}(x^{(j)}) \). This means that \( \tilde{\varphi}_t \) can be maximised by maximising each \( \tilde{\varphi}_t^{(j)} \) separately on \( X^{(j)} \). As we need to solve \( M \) at most \( d \) dimensional optimisation problems, it requires only \( O(Md+1\zeta^{-d}) \) calls to the utility function in total – far more favourable than maximising \( \varphi_t \).

Since the cost for maximising the acquisition function is a key theme in this paper let us delve into this a bit more. One call to \( \varphi_t \) requires \( O(Dt^2) \) effort. For \( \tilde{\varphi}_t \) we need \( M \) calls each requiring \( O(dt^2) \) effort. So both \( \varphi_t \) and \( \tilde{\varphi}_t \) require the same effort in this front. For \( \tilde{\varphi}_t \), we need to know the posterior for only \( f \) whereas for \( \tilde{\varphi}_t \) we need to know the posterior for each \( f^{(j)} \). However, the brunt of the work in obtaining the posterior is the \( O(t^3) \) effort in inverting the \( t \times t \) matrix \( \Delta \) in (5) which needs to be done for both \( \varphi_t \) and \( \tilde{\varphi}_t \). For \( \tilde{\varphi}_t \), we can obtain the inverse once and reuse it \( M \) times, so the cost of obtaining the posterior is \( O(t^3+Mt^2) \).

Since the number of queries needed will be super linear in \( D \) and hence \( M \), the \( t^3 \) term dominates. Therefore obtaining each posterior \( f^{(j)} \) is only marginally more work than

---

**Figure 1.** Illustration of the additive GP model for 2 observations where \( M = 2 \) in (1). The squared variables are observed while the circled variables are not. For brevity we have denoted \( f_1^{(j)} = f^{(j)}(x_1^{(j)}) \) for \( i = 1, 2, \ast \). We wish to infer the posterior distributions of the individual GPs \( f^{(j)}(x_1^{(j)}) \) (outlined in blue).
obtaining the posterior for \( f \). Any difference here is easily offset by the cost for maximising the acquisition function.

The question remains then if maximising \( \tilde{\varphi}_t \) would result in low regret. Since \( \varphi_t \) and \( \tilde{\varphi}_t \) are neither equivalent nor have the same maximiser it is not immediately apparent that this should work. Nonetheless, intuitively this seems like a reasonable scheme since the \( \sum_j \sigma_j^2 \) term captures some notion of the uncertainty and contributes to exploration. In Theorem 5 we show that this intuition is reasonable – maximising \( \tilde{\varphi}_t \) achieves the same rates as \( \varphi_t \) for cumulative and simple regrets if the kernel is additive.

We summarise the resulting algorithm in Algorithm 1. In brief, at time step \( t \), we obtain the posterior distribution for \( f^{(j)} \) and maximise \( \tilde{\varphi}_t^{(j)} \) to determine the coordinates \( x_t^{(j)} \).

We do this for each \( j \) and then combine them to obtain \( x_t \).

Algorithm 1 Add-GP-UCB

**Input:** Kernels \( \kappa^{(1)}, \ldots, \kappa^{(M)} \), Decomposition \( (X^{(j)}(U))_{j=1}^{M} \)

1. \( D_0 = \emptyset \),
2. for \( j = 1, \ldots, M \), \( (\mu_0^{(j)}, \kappa_0^{(j)}) \leftarrow (0, \kappa^{(j)}) \),
3. for \( t = 1, 2, \ldots \)
   1. for \( j = 1, \ldots, M \), \( x_t^{(j)} \leftarrow \arg \max_{z \in X^{(j)}} \mu_t^{(j)}(z) + \sqrt{d_t} \sigma_t^{(j)}(z) \),
   2. \( x_t \leftarrow \bigcup_{j=1}^{M} x_t^{(j)} \),
   3. \( y_t \leftarrow \text{Query at } x_t \),
   4. \( D_t = D_{t-1} \cup \{x_t, y_t\} \),
   5. Perform Bayesian posterior updates conditioned on \( D_t \) to obtain \( \mu_t^{(j)}, \sigma_t^{(j)} \) for each \( j = 1, \ldots, M \).

4.3. Main Theoretical Results

Now, we present our main theoretical contributions. We bound the regret for Add-GP-UCB under different kernels. Following Srinivas et al. (2010), we first bound the statistical difficulty of the problem as determined by the kernel. We show that under additive kernels the problem is much easier than when using a full \( D^b \) order kernel. Next, we show that the Add-GP-UCB algorithm is able to exploit the additive structure and obtain the same rates as GP-UCB. The advantage to using Add-GP-UCB is that it is much easier to optimise the acquisition function. For our analysis, we will need Assumption 2 and Definition 3.

**Assumption 2.** Let \( f \) be sampled from a GP with kernel \( \kappa \), \( \kappa(\cdot, x) \) is L-Lipschitz for all \( x \). Further, the partial derivatives of \( f \) satisfies the following high probability bound. There exists constants \( a, b > 0 \) such that,

\[
\mathbb{P} \left( \sup_x \left| \frac{\partial f(x)}{\partial x_t} \right| > J \right) \leq ae^{-(J/b)^2}.
\]

The Lipschitzian condition is fairly mild and the latter condition holds for four times differentiable stationary kernels such as the SE and Matérn kernels for \( \nu > 2 \) (Ghosal & Roy, 2006). Srinivas et al. (2010) showed that the statistical difficulty of GP/B/BO is determined by the Maximum Information Gain as defined below. We bound this quantity for additive SE and Matérn kernels in Theorem 4. This is our first main theorem.

**Definition 3.** (Maximum Information Gain) Let \( f \sim GP(\mu, \kappa) \), \( y_t = f(x_t) + \epsilon \) where \( \epsilon \sim N(0, \eta^2) \). Let \( A = \{x_1, \ldots, x_T\} \subset X \) be a finite subset, \( f_A \) denote the function values at these points and \( y_A \) denote the noisy observations. Let \( I \) be the Shannon Mutual Information. The Maximum Information Gain between \( y_A \) and \( f_A \) is

\[
\gamma_T = \max_{\mathcal{A} \subset X, |\mathcal{A}|=T} I(y_A; f_A).
\]

**Theorem 4.** Assume that the kernel \( \kappa \) has the additive form of (4), and that each \( \kappa^{(j)} \) satisfies Assumption 2. W.l.o.g. assume \( \kappa(x, x') = 1 \), Then,

1. If each \( \kappa^{(j)} \) is a \( d_j^b \) order squared exponential kernel (2) where \( d_j \leq d \), then \( \gamma_T \in \mathcal{O}(Dd^{d}(\log T)^{d+1}) \).
2. If each \( \kappa^{(j)} \) is a \( d_j^b \) order Matérn kernel (3) where \( d_j \leq d \) and \( \nu > 2 \), then \( \gamma_T \in \mathcal{O}(D^{2d^{d}(\log T)}) \).

The proof is given in Appendix B.1. The important observation is that the dependence on \( D \) is linear for an additive kernel. In contrast, for a \( D^b \) order kernel this is exponential (Srinivas et al., 2010). Next, we present our second main theorem which bounds the regret for Add-GP-UCB for an additive kernel as given in Equation 4.

**Theorem 5.** Suppose \( f \) is constructed by sampling \( f^{(j)} \sim GP(0, \kappa^{(j)}) \) for \( j = 1, \ldots, M \) and then adding them. Let all kernels \( \kappa^{(j)} \) satisfy assumption 2 for some \( L, a, b \). Further, we maximise the acquisition function \( \tilde{\varphi}_t \) to within \( \zeta_0 t^{-1/2} \) accuracy at time step \( t \). Pick \( \delta \in (0, 1) \) and choose

\[
\beta_t = 2 \log \left( \frac{M \pi^2 t^2}{2\delta} \right) + 2d \log \left( D d^{d} \right) \in \mathcal{O}(d \log T).
\]

Then, Add-GP-UCB attains cumulative regret \( R_T \in \mathcal{O} \left( \sqrt{D \gamma_T T \log T} \right) \) and hence simple regret \( S_T \in \mathcal{O} \left( \sqrt{D \gamma_T T \log T} \right) \). Precisely, with probability \( > 1 - \delta \),

\[
\forall t \geq 1, \quad R_T \leq \sqrt{SC_1 \beta_T MT \gamma_t + 2C_2 \sqrt{T} + C_2}.
\]

where \( C_1 = 1/\log(1 + \eta^{-2}) \) and \( C_2 \) is a constant depending on \( a, b, D, \delta, L \) and \( \eta \).

Our proof uses ideas from Srinivas et al. (2010). We also show that complete maximisation of \( \tilde{\varphi}_t \) is not required provided that the accuracy improves at rate \( O(t^{1/2}) \). The proof
is given in Appendix B.2. When we combine the results in Theorems 4 and 5 we obtain the rates given in Table 1.

One could consider alternative lower order kernels – one candidate is the sum of all possible $d^{th}$ order kernels (Duvenaud et al., 2011). Such a kernel would arguably allow us to represent a larger class of functions than our kernel in (4). If, for instance, we choose each of them to be a SE kernel, then it can be shown that $\gamma_p \in O(D^d d^d + (\log T)^{d+1})$. Even though this is worse than our kernel in poly($D$) factors, it is still substantially better than using a $D^n$ order kernel. However, maximising the corresponding utility function, either of the form $\varphi_t$ or $\tilde{\varphi}_t$, is still a $D$ dimensional problem. We reiterate that what renders our algorithm attractive in large $D$ is not just the statistical gains due to the simpler kernel. It is also the fact that our acquisition function can be efficiently maximised.

4.4. Practical Considerations

Our practical implementation differs from our theoretical analysis in the following aspects.

**Choice of $\beta$:** $\beta_t$ as specified by Theorems 5, usually tends to be conservative in practice (Srinivas et al., 2010). For good empirical performance a more aggressive strategy is required. In our experiments, we set $\beta_t = 0.2d \log(2t)$ which offered a good tradeoff between exploration and exploitation. Here $d$ is the dimension of the space in which we are optimising the acquisition. Note that this captures the correct dependence on $D, d$ and $t$ in Theorems 5 and 6.

**Data dependent prior:** Our analysis assumes that we know the GP kernel of the prior. In reality this is rarely the case. In our experiments, we choose the hyperparameters of the kernel by maximising the GP marginal likelihood (Rasmussen & Williams, 2006) every $N_{\text{cyc}}$ iterations.

**Initialisation:** Marginal likelihood based kernel tuning can be unreliable with few data points. This is a problem in the first few iterations. Following the recommendations in Bull (2011) we initialise **Add-GP-UCB** (and **GP-UCB**) using $N_{\text{init}}$ points selected uniformly at random.

**Decomposition & Non-additive functions:** If $f$ is additive and the decomposition is known, we use it directly. But it may not always be known or $f$ may not be additive. Then, we could treat the decomposition as a hyperparameter of the additive kernel and maximise the marginal likelihood w.r.t the decomposition. However, given that there are $D! / d! M!$ possible decompositions, computing the marginal likelihood for all of them is infeasible. We circumvent this issue by randomly selecting a few $(O(D))$ decompositions and choosing the one with the largest marginal likelihood. Intuitively, if the function is not additive, with such a “partial maximisation” we can hope to capture some existing marginal structure in $f$. At the same time, even an exhaustive maximisation will not do much better than a partial maximisation if there is no additive structure. Empirically, we found that partially optimising for the decomposition performed slightly better than using a fixed decomposition or a random decomposition at each step. We incorporate this procedure for finding an appropriate decomposition as part of the kernel hyperparameter learning procedure every $N_{\text{cyc}}$ iterations.

How do we choose $(d, M)$ when $f$ is not additive? If $d$ is large we allow for richer class of functions, but risk high variance. For small $d$, the kernel is too simple and we have high bias but low variance – further optimising $\tilde{\varphi}_t$ is easier. In practice we found that our procedure was fairly robust for reasonable choices of $d$. Yet this is an interesting theoretical question. We also believe it is a difficult one. Using the marginal likelihood alone will not work as the optimal choice of $d$ also depends on the computational budget for optimising $\tilde{\varphi}_t$. We hope to study this question in future work. For now, we give some recommendations at the end. Our modified algorithm with these practical considerations is given below. Observe that in this specification if we use $d = D$ we have the original **GP-UCB** algorithm.

**Algorithm 2 Practical-Add-GP-UCB**

Input: $N_{\text{init}}, N_{\text{cyc}}, d, M$

- $D_0 \leftarrow N_{\text{init}}$ points chosen uniformly at random.
- for $t = 1, 2, \ldots$
  1. if $(t \mod N_{\text{cyc}} = 0)$, Learn the kernel hyperparameters and the decomposition $\{X_j\}$ by maximising the GP marginal likelihood.
  2. Perform steps 1-3 in Algorithm 1 with $\beta_t = 0.2d \log 2t$.
  3. $D_t = D_{t-1} \cup \{ (x_t, y_t) \}$.
  4. Perform Bayesian posterior updates conditioned on $D_t$ to obtain $\mu_t^{(j)}$, $\sigma_t^{(j)}$ for $j = 1, \ldots, M$.

5. Summary of Experiments

In this summary we present results in the optimisation setting. Refer Appendix C for results on bandits. Following Brochu et al. (2010) we use DiRect to maximise $\varphi_t$, $\tilde{\varphi}_t$. To demonstrate the efficacy of **Add-GP-UCB** we optimise the acquisition function under a constrained budget. We compare **Add-GP-UCB** against **GP-UCB**, random querying (RAND) and DiRect. On the real datasets we also compare it to the Expected Improvement (GP-EI) acquisition function which is popular in BO applications and the method of Wang et al. (2013) which uses a random projection before applying BO (REMBO). We have multiple instantiations of **Add-GP-UCB** for different values for $(d, M)$.

In contrast to existing literature in the BO community,
Additive Gaussian Process Optimisation and Bandits

Figure 2. Results on the synthetic experiments. The x-axis is the number of queries and the y-axis is the regret in log scale. We have indexed the experiments by their \((D, d', M')\) values. In some figures, the error bars are not visible since they are small and hidden by the bullets. All figures were produced by averaging over 20 runs.

we found that the UCB acquisitions outperformed GP-EI. One possible reason may be that under a constrained budget, UCB is robust to imperfect maximisation (Theorem 5) whereas GP-EI may not be. Another reason may be our choice of constants in UCB (Section 4.4).

### 5.1. Simulations on Synthetic Functions

We create a series of additive functions by replicating a \(d'\) dimensional function \(f_{d'}\) in \(M'\) groups. (We use the prime to avoid confusion with our Add-GP-UCB instantiations with different \((d, M)\) values.) So the function doesn’t depend on \(D - d'M'\) coordinates. We have illustrated \(f_{d'}\) for \(d' = 2\) in the first figure in Fig 2 (See Eq (14) in C.1). Since each \(f_{d'}\) has 3 modes, the function has \(3^M'\) modes. In the synthetic experiments we use an instantiation of Add-GP-UCB that knows the decomposition—i.e. \((d, M) = (d', M')\) and the grouping of coordinates. We refer to this as Add-*. For the rest we use a \((d, M)\) decomposition by creating \(M\) groups of size at most \(d\) and find a good grouping by partially maximising the marginal likelihood (Section 4.4). We refer to them as Add-\(d/M\).

For GP-UCB and GP-EI we allocate a budget of \(\min(5000, 100D)\) DiRect function evaluations to optimise the acquisition function. For all Add-\(d/M\) methods we set it to 90% of this amount to account for the additional overhead in posterior inference for each \(f^{(j)}\). While the 90% seems arbitrary, in our experiments this was hardly a factor as the cost was dominated by the inversion of \(\Delta\). Therefore, for our 10D problem we maximise \(\varphi_t\) with \(\beta_t = 2\log(2t)\) with 1000 evaluations whereas for Add-5/2 we maximise each \(\varphi_t^{(j)}\) with \(\beta_t = \log(2t)\) with 450 evaluations.

We refer to each example by the configuration of the additive function—its \((D, d', M')\) values. In the \((10, 3, 3)\) example Add-* does best since it knows the correct model and the acquisition function can be maximised within the budget. However Add-3/4 and Add-5/2 models do well too and outperform GP-UCB. Add-1/10 performs poorly since it is statistically not expressive enough to capture the true function (high bias). In the \((24, 11, 2), (40, 18, 2)\) and \((96, 29, 3)\) examples Add-* outperforms GP-UCB. However, it is not competitive with the Add-\(d/M\) for small \(d\). Even though Add-* knows the correct decomposition, there are two possible failure modes since \(d'\) is large. The variance is very high in the absence of sufficient data points. In addition, optimising the acquisition function is also difficult. This illustrates our previous argument that using an additive kernel can be advantageous even on non-additive functions. In the \((40, 5, 8), (96, 5, 19)\) examples Add-* performs best as \(d'\) is small enough. But again, almost all Add-\(d/M\) instantiations outperform GP-UCB. In contrast to the small \(D\) examples, for large \(D\), GP-UCB and Add-\(d/M\) with large \(d\) perform worse than DiRect. This is probably because the acquisition cannot be maximised to sufficient accuracy within the budget. We have
5.2. Real Experiments

SDSS Galaxy Data: Here, we use galaxy data from the Sloan Digital Sky Survey to find the maximum likelihood values for 20 cosmological parameters. The likelihood is computed via an astrophysical simulation. Software is obtained from Tegmark et al. (2006). Each query to the likelihood takes 2-5 seconds. The likelihood only depends on 9 of the parameters but we augment it to 20 dimensions to emulate the fact that in real astrophysical applications we may not know the relevant parameters. In order to be wall clock time competitive with RAND and DiRect we use 500 evaluations for GP-UCB. GP-EI and REMBO and 450 for Add-\(d/M\) to maximise the acquisition function. We have elaborated more details in Appendix C.2. The results are given in 3(a). Despite the fact that the function may not be additive, all Add-\(d/M\) methods outperform GP-UCB and GP-EI. Since the function only depends on 9 parameters we used REMBO with a 9 dimensional projection. Despite this advantage to REMBO it is not as competitive with the Add-\(d/M\) methods. Here Add-5/4 performs slightly better than the rest since it seems to have the best tradeoff between being statistically expressive enough to capture the function while at the same time being easy enough to optimise the acquisition function within the allocated budget.

Viola & Jones Face Detection: The Viola & Jones Cascade Classifier (VJ) (Viola & Jones, 2001) is a popular method for face detection in computer vision based on the Adaboost algorithm. In this experiment we use the VJ face dataset and the OpenCV implementation (Bradski & Kaehler, 2008) which implements the classifier as a 22-stage cascade. The task is to find the 22 threshold values for each stage to maximise classification accuracy. Each function call takes 30-40 seconds and is the dominant cost in this experiment. We use 1000 DiRect evaluations to optimise the acquisition function for GP-UCB, GP-EI and REMBO and 900 for Add-\(d/M\). We use REMBO with a 5 dimensional projection. The results are given in Figure 3(b). Not surprisingly, REMBO performs worst since it is searching only on a 5 dimensional space. Barring Add-1/22 all other Add-\(d/M\) instantiations outperform GP-UCB and GP-EI with Add-6/4 performing best. Interestingly, we also find a configuration for the thresholds that outperforms the one used in OpenCV.

6. Conclusion

Recommendations: Based on our experiences, we recommend the following. If \(f\) is known to be additive, the decomposition is known and \(d'\) is small enough so that \(\varphi_i\) can be efficiently optimised, then running Add-GP-UCB with the known decomposition is likely to produce the best results. If not, then use a small value for \(d\) and run Add-GP-UCB while partially optimising for the decomposition periodically (Section 4.4). In our experiments we found that using \(d\) between 3 an 12 seemed reasonable choices. However, note that this depends on the computational budget for optimising the acquisition, the query budget for \(f\) and to a certain extent the the function \(f\) itself.

Summary: Our algorithm takes into account several practical considerations in real world GPB/BO applications such as computational constraints in optimising the acquisition and the fact that we have to work with a relatively few data points since function evaluations are expensive. Our framework effectively addresses these concerns without considerably compromising on the statistical integrity of the model. We believe that this provides a promising direction to scale GPB/BO methods to high dimensions.

Future Work: Our experiments indicate that our methods perform well beyond the scope suggested by our theory. Developing an analysis that takes into account the bias-variance and computational tradeoffs in approximating and optimising a non-additive function via an additive model is an interesting challenge. We also intend to extend this framework to other acquisition functions.
Acknowledgements

We wish to thank Akshay Krishnamurthy and Andrew Gordon Wilson for the insightful discussions and Andreas Krause, Sham Kakade and Matthias Seeger for the helpful email conversations. This research is partly funded by DOE grant DESC0011114.

References


Thompson, W. R. On the Likelihood that one Unknown Probability Exceeds Another in View of the Evidence of Two Samples. *Biometrika*, 1933.

