Max-value Entropy Search for Efficient Bayesian Optimization

Zi Wang [1] **Stefanie Jegelka** [1]

Abstract

Entropy Search (ES) and Predictive Entropy Search (PES) are popular and empirically successful Bayesian Optimization techniques. Both rely on a compelling information-theoretic motivation, and maximize the information gained about the $\arg\max$ of the unknown function; yet, both are plagued by the expensive computation for estimating entropies. We propose a new criterion, Max-value Entropy Search (MES), that instead uses the information about the maximum function value. We show relations of MES to other Bayesian optimization methods, and establish a regret bound. We observe that MES maintains or improves the good empirical performance of ES/PES, while tremendously lightening the computational burden. In particular, MES is much more robust to the number of samples used for computing the entropy, and hence more efficient for higher dimensional problems.

1. Introduction

Bayesian optimization (BO) has become a popular and effective way for black-box optimization of nonconvex, expensive functions in robotics, machine learning, computer vision, and many other areas of science and engineering (Brochu et al., 2009; Calandra et al., 2014; Krause & Ong, 2011; Lizotte et al., 2007; Snoek et al., 2012; Thornton et al., 2013; Wang et al., 2017). In BO, a prior is posed on the (unknown) objective function, and the uncertainty given by the associated posterior is the basis for an acquisition function that guides the selection of the next point to query the function. The selection of queries and hence the acquisition function is critical for the success of the method.

Different BO techniques differ in this acquisition function. Among the most popular ones range the Gaussian process upper confidence bound (GP-UCB) (Auer, 2002; Srinivas et al., 2010), probability of improvement (PI) (Kushner, 1964), and expected improvement (EI) (Močkus, 1974). Particularly successful recent additions are entropy search (ES) (Hennig & Schuler, 2012) and predictive entropy search (PES) (Hernández-Lobato et al., 2014), which aim to maximize the mutual information between the queried points and the location of the global optimum.

ES and PES are effective in the sense that they are query-efficient and identify a good point within competitively few iterations, but determining the next query point involves very expensive computations. As a result, these methods are most useful if the black-box function requires a lot of effort to evaluate, and are relatively slow otherwise. Moreover, they rely on estimating the entropy of the $\arg\max$ of the function. In high dimensions, this estimation demands a large number of samples from the input space, which can quickly become inefficient.

We propose a twist to the viewpoint of ES and PES that retains the information-theoretic motivation and empirically successful query-efficiency of those methods, but at a much reduced computational cost. The key insight is to replace the uncertainty about the $\arg\max$ with the uncertainty about the maximum function value. As a result, we refer to our new method as *Max-value Entropy Search (MES)*. As opposed to the $\arg\max$, the maximum function value lives in a one-dimensional space, which greatly facilitates the estimation of the mutual information via sampling. We explore two strategies to make the entropy estimation efficient: an approximation by a Gumbel distribution, and a Monte Carlo approach that uses random features.

Our contributions are as follows: (1) MES, a variant of the entropy search methods, which enjoys efficient computation and simple implementation; (2) an intuitive analysis which establishes the first connection between ES/PES and the previously proposed criteria GP-UCB, PI and EST (Wang et al., 2016), where the bridge is formed by MES; (3) a regret bound for a variant of MES, which, to our knowledge, is the first regret bound established for any variant of the entropy search methods; (4) an extension of MES to the high dimensional settings via additive Gaussian processes; and (5) empirical evaluations which demon-

[1]Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Massachusetts, USA. Correspondence to: Zi Wang <ziw@csail.mit.edu>, Stefanie Jegelka <stefje@csail.mit.edu>.

Proceedings of the 34^{th} International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017.

strate that MES identifies good points as quickly or better than ES/PES, but is much more efficient and robust in estimating the mutual information, and therefore much faster than its input-space counterparts.

After acceptance of this work, we learned that Hoffman & Ghahramani (2015) independently arrived at the acquisition function in Eq. (5). Yet, our approximation (Eq. (6)) is different, and hence the actual acquisition function we evaluate and analyze is different.

2. Background

Our goal is to maximize a black-box function $f : \mathfrak{X} \to \mathbb{R}$ where $\mathfrak{X} \subset \mathbb{R}^d$ and $\mathfrak{X}$ is compact. At time step t, we select point $\boldsymbol{x}_t$ and observe a possibly noisy function evaluation $y_t = f(\boldsymbol{x}_t) + \epsilon_t$, where $\epsilon_t \sim \mathcal{N}(0, \sigma^2)$ are i.i.d. Gaussian variables. We use Gaussian processes (Rasmussen & Williams, 2006) to build a probabilistic model of the black-box function to be optimized. For high dimensional cases, we use a variant of the additive Gaussian process (Duvenaud et al., 2011; Kandasamy et al., 2015). For completeness, we here introduce some basics of GP and add-GP.

2.1. Gaussian Processes

Gaussian processes (GPs) are distributions over functions, and popular priors for Bayesian nonparametric regression. In a GP, any finite set of function values has a multivariate Gaussian distribution. A Gaussian process $GP(\mu, k)$ is fully specified by a mean function $\mu(\boldsymbol{x})$ and covariance (kernel) function $k(\boldsymbol{x}, \boldsymbol{x}')$. Let f be a function sampled from $GP(\mu, k)$. Given the observations $D_t = \{(\boldsymbol{x}_\tau, y_\tau)\}_{\tau=1}^t$, we obtain the posterior mean $\mu_t(\boldsymbol{x}) = \boldsymbol{k}_t(\boldsymbol{x})^{\mathrm{T}}(\boldsymbol{K}_t + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{y}_t$ and posterior covariance $k_t(\boldsymbol{x}, \boldsymbol{x}') = k(\boldsymbol{x}, \boldsymbol{x}') - \boldsymbol{k}_t(\boldsymbol{x})^{\mathrm{T}}(\boldsymbol{K}_t + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{k}_t(\boldsymbol{x}')$ of the function via the kernel matrix $\boldsymbol{K}_t = [k(\boldsymbol{x}_i, \boldsymbol{x}_j)]_{\boldsymbol{x}_i, \boldsymbol{x}_j \in D_t}$ and $\boldsymbol{k}_t(\boldsymbol{x}) = [k(\boldsymbol{x}_i, \boldsymbol{x})]_{\boldsymbol{x}_i \in D_t}$ (Rasmussen & Williams, 2006). The posterior variance is $\sigma_t^2(\boldsymbol{x}) = k_t(\boldsymbol{x}, \boldsymbol{x})$.

2.2. Additive Gaussian Processes

Additive Gaussian processes (add-GP) were proposed in (Duvenaud et al., 2011), and analyzed in the BO setting in (Kandasamy et al., 2015). Following the latter, we assume that the function f is a sum of independent functions sampled from Gaussian processes that are active on disjoint sets A_m of input dimensions. Precisely, $f(x) = \sum_{m=1}^M f^{(m)}(x^{A_m})$, with $A_i \cap A_j = \emptyset$ for all $i \neq j$, $|\cup_{i=1}^M A_i| = d$, and $f^{(m)} \sim GP(\mu^{(m)}, k^{(m)})$, for all $m \leq M$ ($M \leq d < \infty$). As a result of this decomposition, the function f is distributed according to $GP(\sum_{m=1}^M \mu^{(m)}, \sum_{m=1}^M k^{(m)})$. Given a set of noisy observations $D_t = \{(\boldsymbol{x}_\tau, y_\tau)\}_{\tau=1}^t$ where $y_\tau \sim \mathcal{N}(f(x_\tau), \sigma^2)$, the posterior mean and covariance of the function component $f^{(m)}$ can be inferred as $\mu_t^{(m)}(\boldsymbol{x}) = \boldsymbol{k}_t^{(m)}(\boldsymbol{x})^{\mathrm{T}}(\boldsymbol{K}_t + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{y}_t$ and $k_t^{(m)}(\boldsymbol{x}, \boldsymbol{x}') = k^{(m)}(\boldsymbol{x}, \boldsymbol{x}') - \boldsymbol{k}_t^{(m)}(\boldsymbol{x})^{\mathrm{T}}(\boldsymbol{K}_t + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{k}_t^{(m)}(\boldsymbol{x}')$, where $\boldsymbol{k}_t^{(m)}(\boldsymbol{x}) = [k^{(m)}(\boldsymbol{x}_i, \boldsymbol{x})]_{\boldsymbol{x}_i \in D_t}$ and $\boldsymbol{K}_t = \left[\sum_{m=1}^M k^{(m)}(\boldsymbol{x}_i, \boldsymbol{x}_j)\right]_{\boldsymbol{x}_i, \boldsymbol{x}_j \in D_t}$. For simplicity, we use the shorthand $k^{(m)}(\boldsymbol{x}, \boldsymbol{x}') = k^{(m)}(\boldsymbol{x}^{A_m}, \boldsymbol{x}'^{A_m})$.

2.3. Evaluation Criteria

We use two types of evaluation criteria for BO, *simple regret* and *inference regret*. In each iteration, we choose to evaluate one input $\boldsymbol{x}_t$ to "learn" where the $\arg\max$ of the function is. The simple regret $r_T = \max_{\boldsymbol{x} \in \mathfrak{X}} f(\boldsymbol{x}) - \max_{t \in [1,T]} f(\boldsymbol{x}_t)$ measures the value of the best queried point so far. After all queries, we may infer an $\arg\max$ of the function, which is usually chosen as $\tilde{\boldsymbol{x}}_T = \arg\max_{\boldsymbol{x} \in \mathfrak{X}} \mu_T(\boldsymbol{x})$ (Hennig & Schuler, 2012; Hernández-Lobato et al., 2014). We denote the inference regret as $R_T = \max_{\boldsymbol{x} \in \mathfrak{X}} f(\boldsymbol{x}) - f(\tilde{x}_T)$ which characterizes how satisfying our inference of the $\arg\max$ is.

3. Max-value Entropy Search

Entropy search methods use an information-theoretic perspective to select where to evaluate. They find a query point that maximizes the information about the location $\boldsymbol{x}_* = \arg\max_{\boldsymbol{x} \in \mathfrak{X}} f(x)$ whose value $y_* = f(\boldsymbol{x}_*)$ achieves the global maximum of the function f. Using the negative differential entropy of $p(\boldsymbol{x}_*|D_t)$ to characterize the uncertainty about $\boldsymbol{x}_*$, ES and PES use the acquisition functions

$$\alpha_t(x) = I(\{\boldsymbol{x}, y\}; \boldsymbol{x}_* \mid D_t) \tag{1}$$
$$= H\left(p(\boldsymbol{x}_* \mid D_t)\right) - \mathbb{E}\left[H(p(\boldsymbol{x}_* \mid D_t \cup \{\boldsymbol{x}, y\}))\right] \tag{2}$$
$$= H(p(y \mid D_t, \boldsymbol{x})) - \mathbb{E}\left[H(p(y \mid D_t, \boldsymbol{x}, \boldsymbol{x}_*))\right]. \tag{3}$$

ES uses formulation (2), in which the expectation is over $p(y|D_t, \boldsymbol{x})$, while PES uses the equivalent, symmetric formulation (3), where the expectation is over $p(\boldsymbol{x}_*|D_t)$. Unfortunately, both $p(\boldsymbol{x}_*|D_t)$ and its entropy is analytically intractable and have to be approximated via expensive computations. Moreover, the optimum may not be unique, adding further complexity to this distribution.

We follow the same information-theoretic idea but propose a much cheaper and more robust objective to compute. Instead of measuring the information about the argmax $\boldsymbol{x}_*$, we use the information about the *maximum value* $y_* = f(\boldsymbol{x}_*)$. Our acquisition function is the gain in mutual information between the maximum y_* and the next point we query, which can be approximated analytically by evaluating the entropy of the predictive distribution:

$$\alpha_t(x) = I(\{\boldsymbol{x}, y\}; y_* \mid D_t) \tag{4}$$
$$= H(p(y \mid D_t, \boldsymbol{x})) - \mathbb{E}[H(p(y \mid D_t, \boldsymbol{x}, y_*))] \tag{5}$$

$$\approx \frac{1}{K} \sum_{y_* \in Y_*} \left[\frac{\gamma_{y_*}(\boldsymbol{x})\psi(\gamma_{y_*}(\boldsymbol{x}))}{2\Psi(\gamma_{y_*}(\boldsymbol{x}))} - \log(\Psi(\gamma_{y_*}(\boldsymbol{x}))) \right] \quad (6)$$

where ψ is the probability density function and Ψ the cumulative density function of a normal distribution, and $\gamma_{y_*}(\boldsymbol{x}) = \frac{y_* - \mu_t(\boldsymbol{x})}{\sigma_t(\boldsymbol{x})}$. The expectation in Eq. (5) is over $p(y_*|D_n)$, which is approximated using Monte Carlo estimation by sampling a set of K function maxima. Notice that the probability in the first term $p(y|D_t, \boldsymbol{x})$ is a Gaussian distribution with mean $\mu_t(\boldsymbol{x})$ and variance $k_t(\boldsymbol{x}, \boldsymbol{x})$. The probability in the second term $p(y|D_n, \boldsymbol{x}, y_*)$ is a truncated Gaussian distribution: given y_*, the distribution of y needs to satisfy $y < y_*$. Importantly, while ES and PES rely on the expensive, d-dimensional distribution $p(\boldsymbol{x}_*|D_t)$, here, we use the one-dimensional $p(y_*|D_n)$, which is computationally much easier.

It may not be immediately intuitive that the *value* should bear sufficient information for a good search strategy. Yet, the empirical results in Section 5 will demonstrate that this strategy is typically at least as good as ES/PES. From a formal perspective, Wang et al. (2016) showed how an estimate of the maximum value implies a good search strategy (EST). Indeed, Lemma 3.1 will make the relation between EST and a simpler, degenerate version of MES explicit.

Hence, it remains to determine how to sample y_*. We propose two strategies: (1) sampling from an approximation via a Gumbel distribution; and (2) sampling functions from the posterior Gaussian distribution and maximizing the functions to obtain samples of y_*. We present the MES algorithm in Alg. 1.

Algorithm 1 Max-value Entropy Search (MES)

function MES (f, D_0)
  for $t = 1, \cdots, T$ **do**
    $\alpha_{t-1}(\cdot) \leftarrow$ APPROX-MI (D_{t-1})
    $\boldsymbol{x}_t \leftarrow \arg\max_{\boldsymbol{x} \in \mathfrak{X}} \alpha_{t-1}(\boldsymbol{x})$
    $y_t \leftarrow f(\boldsymbol{x}_t) + \epsilon_t, \epsilon_t \sim \mathcal{N}(0, \sigma^2)$
    $\mathfrak{D}_t \leftarrow D_{t-1} \cup \{\boldsymbol{x}_t, y_t\}$
  end for
end function

function Approx-MI (D_t)
  if Sample with Gumbel **then**
    approximate $\Pr[\hat{y}_* < y]$ with $\mathcal{G}(a, b)$
    sample a K-length vector $\boldsymbol{r} \sim$ Unif$([0, 1])$
    $\boldsymbol{y}_* \leftarrow a - b \log(-\log \boldsymbol{r})$
  else
    for $i = 1, \cdots, K$ **do**
      sample $\tilde{f} \sim GP(\mu_t, k_t \mid D_t)$
      $y_{*(i)} \leftarrow \max_{\boldsymbol{x} \in \mathfrak{X}} \tilde{f}(\boldsymbol{x})$
    end for
    $\boldsymbol{y}_* \leftarrow [y_{*(i)}]_{i=1}^K$
  end if
  return $\alpha_t(\cdot)$ in Eq. (6)
end function

3.1. Gumbel Sampling

The marginal distribution of $f(x)$ for any x is a one-dimensional Gaussian, and hence the distribution of y^* may be viewed as the maximum of an infinite collection of dependent Gaussian random variables. Since this distribution is difficult to compute, we make two simplifications. First, we replace the continuous set $\mathfrak{X}$ by a discrete (finite), dense subset $\hat{\mathfrak{X}}$ of representative points. If we select $\hat{\mathfrak{X}}$ to be an ϵ-cover of $\mathfrak{X}$ and the function f is Lipschitz continuous with constant L, then we obtain a valid upper bound on $f(\mathfrak{X})$ by adding ϵL to any upper bound on $f(\hat{\mathfrak{X}})$.

Second, we use a "mean field" approximation and treat the function values at the points in $\hat{\mathfrak{X}}$ as independent. This approximation tends to over-estimate the maximum; this follows from Slepian's lemma if $k(x, x') \geq 0$. Such upper bounds still lead to optimization strategies with vanishing regret, whereas lower bounds may not (Wang et al., 2016).

We sample from the approximation $\hat{p}(y^*|D_n)$ via its cumulative distribution function (CDF) $\widehat{\Pr}[y_* < z] = \prod_{\boldsymbol{x} \in \hat{\mathfrak{X}}} \Psi(\gamma_z(\boldsymbol{x}))$. That means we sample r uniformly from $[0, 1]$ and find z such that $\Pr[y_* < z] = r$. A binary search for z to accuracy δ requires $O(\log \frac{1}{\delta})$ queries to the CDF, and each query takes $O(|\hat{\mathfrak{X}}|) \approx O(n^d)$ time, so we obtain an overall time of $O(M|\hat{\mathfrak{X}}| \log \frac{1}{\delta})$ for drawing M samples.

To sample more efficiently, we propose a $O(M + |\hat{\mathfrak{X}}| \log \frac{1}{\delta})$-time strategy, by approximating the CDF by a Gumbel distribution: $\widehat{\Pr}[y_* < z] \approx \mathcal{G}(a, b) = e^{-e^{-\frac{z-a}{b}}}$. This choice is motivated by the Fisher-Tippett-Gnedenko theorem (Fisher, 1930), which states that the maximum of a set of i.i.d. Gaussian variables is asymptotically described by a Gumbel distribution (see the appendix for further details). This does not in general extend to non-i.i.d. Gaussian variables, but we nevertheless observe that in practice, this approach yields a good and fast approximation.

We sample from the Gumbel distribution via the Gumbel quantile function: we sample r uniformly from $[0, 1]$, and let the sample be $y = \mathcal{G}^{-1}(a, b) = a - b \log(-\log r)$. We set the appropriate Gumbel distribution parameters a and b by percentile matching and solve the two-variable linear equations $a - b \log(-\log r_1) = y_1$ and $a - b \log(-\log r_2) = y_2$, where $\Pr[y_* < y_1] = r_1$ and $\Pr[y_* < y_2] = r_2$. In practice, we use $r_1 = 0.25$ and $r_2 = 0.75$ so that the scale of the approximated Gumbel distribution is proportional to the interquartile range of the CDF $\hat{\Pr}[y_* < z]$.

3.2. Sampling y_* via Posterior Functions

For an alternative sampling strategy we follow (Hernández-Lobato et al., 2014): we draw functions from the posterior GP and then maximize each of the sampled functions. Given the observations $D_t = \{(\boldsymbol{x}_\tau, y_\tau)_{\tau=1}^t\}$, we can approximate the posterior Gaussian process using a 1-hidden-layer neural network $\tilde{f}(\boldsymbol{x}) = \boldsymbol{a}_t^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x})$ where $\boldsymbol{\phi}(x) \in \mathbb{R}^D$ is a vector of feature functions (Neal, 1996; Rahimi et al., 2007) and the Gaussian weight $\boldsymbol{a}_t \in \mathbb{R}^D$ is distributed according to a multivariate Gaussian $\mathcal{N}(\boldsymbol{\nu}_t, \boldsymbol{\Sigma}_t)$.

Computing $\boldsymbol{\phi}(x)$. By Bochner's theorem (Rudin, 2011), the Fourier transform $\hat{k}$ of a continuous and translation-invariant kernel k is guaranteed to be a probability distribution. Hence we can write the kernel of the GP to be $k(\boldsymbol{x}, \boldsymbol{x}') = \mathbb{E}_{\omega\sim\hat{k}(\omega)}[e^{i\omega^{\mathrm{T}}(\boldsymbol{x}-\boldsymbol{x}')}] = \mathbb{E}_{c\sim U[0,2\pi]}\mathbb{E}_{\hat{k}}[2\cos(\omega^{\mathrm{T}}\boldsymbol{x}+c)\cos(\omega^{\mathrm{T}}\boldsymbol{x}'+c)]$ and approximate the expectation by $k(\boldsymbol{x}, \boldsymbol{x}') \approx \boldsymbol{\phi}^{\mathrm{T}}(\boldsymbol{x})\boldsymbol{\phi}(\boldsymbol{x}')$ where $\phi_i(\boldsymbol{x}) = \sqrt{\frac{2}{D}}\cos(\omega_i^{\mathrm{T}}\boldsymbol{x} + c_i)$, $\omega_i \sim \hat{\kappa}(\omega)$, and $c_i \sim U[0, 2\pi]$ for $i = 1, \ldots, D$.

Computing $\boldsymbol{\nu}_t, \boldsymbol{\Sigma}_t$. By writing the GP as a random linear combination of feature functions $\boldsymbol{a}_t^T\boldsymbol{\phi}(\boldsymbol{x})$, we are defining the mean and covariance of the GP to be $\mu_t(\boldsymbol{x}) = \boldsymbol{\nu}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x})$ and $k(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{\phi}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{\Sigma}_t\boldsymbol{\phi}(\boldsymbol{x}')$. Let $Z = [z_1, \cdots, z_t] \in \mathbb{R}^{D\times t}$, where $z_\tau \coloneqq \boldsymbol{\phi}(\boldsymbol{x}_\tau) \in \mathbb{R}^D$. The GP posterior mean and covariance in Section 2.1 become $\mu_t(\boldsymbol{x}) = z^{\mathrm{T}}Z(Z^{\mathrm{T}}Z + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{y}_t$ and $k_t(\boldsymbol{x}, \boldsymbol{x}') = z^{\mathrm{T}}z' - z^{\mathrm{T}}Z(Z^{\mathrm{T}}Z + \sigma^2\boldsymbol{I})^{-1}Z^{\mathrm{T}}z'$. Because $Z(Z^{\mathrm{T}}Z + \sigma^2\boldsymbol{I})^{-1} = (ZZ^{\mathrm{T}} + \sigma^2\boldsymbol{I})^{-1}Z$, we can simplify the above equations and obtain $\boldsymbol{\nu}_t = \sigma^{-2}\boldsymbol{\Sigma}_t Z_t\boldsymbol{y}_t$ and $\boldsymbol{\Sigma}_t = (ZZ^{\mathrm{T}}\sigma^{-2} + \boldsymbol{I})^{-1}$.

To sample a function from this random 1-hidden-layer neural network, we sample $\tilde{\boldsymbol{a}}$ from $\mathcal{N}(\boldsymbol{\nu}_t, \boldsymbol{\Sigma}_t)$ and construct the sampled function $\tilde{f} = \tilde{\boldsymbol{a}}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x})$. Then we optimize $\tilde{f}$ with respect to its input to get a sample of the maximum of the function $\max_{\boldsymbol{x}\in\mathfrak{X}} \tilde{f}(\boldsymbol{x})$.

3.3. Relation to Other BO Methods

As a side effect, our new acquisition function draws connections between ES/PES and other popular BO methods. The connection between MES and ES/PES follows from the information-theoretic viewpoint; the following lemma makes the connections to other methods explicit.

Lemma 3.1. *The following methods are equivalent:*

1. *MES, where we only use a single sample y_* for $\alpha_t(x)$;*
2. *EST with $m = y_*$;*
3. *GP-UCB with $\beta^{\frac{1}{2}} = \min_{\boldsymbol{x}\in\mathfrak{X}} \frac{y_* - \mu_t(\boldsymbol{x})}{\sigma_t(\boldsymbol{x})}$;*
4. *PI with $\theta = y_*$.*

This equivalence no longer holds if we use $M > 1$ samples of y_* in MES.

Proof. The equivalence among 2,3,4 is stated in Lemma 2.1 in (Wang et al., 2016). What remains to be shown is the equivalence between 1 and 2. When using a single y_* in MES, the next point to evaluate is chosen by maximizing $\alpha_t(\boldsymbol{x}) = \gamma_{y_*}(\boldsymbol{x})\frac{\psi(\gamma_{y_*}(\boldsymbol{x}))}{2\Psi(\gamma_{y_*}(\boldsymbol{x}))} - \log(\Psi(\gamma_{y_*}(\boldsymbol{x})))$ and $\gamma_{y_*} = \frac{y_* - \mu_t(\boldsymbol{x})}{\sigma_t(\boldsymbol{x})}$. For EST with $m = y_*$, the next point to evaluate is chosen by minimizing $\gamma_{y_*}(\boldsymbol{x})$. Let us define a function $g(u) = u\frac{\psi(u)}{2\Psi(u)} - \log(\Psi(u))$. Clearly, $\alpha_t(\boldsymbol{x}) = g(\gamma_{y_*}(\boldsymbol{x}))$. Because $g(u)$ is a monotonically decreasing function, maximizing $g(\gamma_{y_*}(\boldsymbol{x}))$ is equivalent to minimizing $\gamma_{y_*}(\boldsymbol{x})$. Hence 1 and 2 are equivalent. □

3.4. Regret Bound

The connection with EST directly leads to a bound on the simple regret of MES, when using only one sample of y_*. We prove Theorem 3.2 in the appendix.

Theorem 3.2. *Let F be the cumulative probability distribution for the maximum of any function f sampled from $GP(\mu, k)$ over the compact search space $\mathfrak{X} \subset R^d$, where $k(\boldsymbol{x}, \boldsymbol{x}') \leq 1, \forall \boldsymbol{x}, \boldsymbol{x}' \in \mathfrak{X}$. Let $f_* = \max_{\boldsymbol{x}\in\mathfrak{X}} f(\boldsymbol{x})$ and $w = F(f_*) \in (0, 1)$, and assume the observation noise is iid $\mathcal{N}(0, \sigma)$. If in each iteration t, the query point is chosen as $\boldsymbol{x}_t = \arg\max_{\boldsymbol{x}\in\mathfrak{X}} \gamma_{y_*^t}(\boldsymbol{x})\frac{\psi(\gamma_{y_*^t}(\boldsymbol{x}))}{2\Psi(\gamma_{y_*^t}(\boldsymbol{x}))} - \log(\Psi(\gamma_{y_*^t}(\boldsymbol{x})))$, where $\gamma_{y_*^t}(\boldsymbol{x}) = \frac{y_*^t - \mu_t(\boldsymbol{x})}{\sigma_t(\boldsymbol{x})}$ and y_*^t is drawn from F, then with probability at least $1 - \delta$, in $T' = \sum_{i=1}^T \log_w \frac{\delta}{2\pi_i}$ number of iterations, the simple regret satisfies*

$$r_{T'} \leq \sqrt{\frac{C\rho_T}{T}}(\nu_{t^*} + \zeta_T) \tag{7}$$

where $C = 2/\log(1 + \sigma^{-2})$ and $\zeta_T = (2\log(\frac{\pi_T}{\delta}))^{\frac{1}{2}}$; π satisfies $\sum_{i=1}^T \pi_i^{-1} \leq 1$ and $\pi_t > 0$, and $t^* = \arg\max_t \nu_t$ with $\nu_t \triangleq \min_{\boldsymbol{x}\in\mathfrak{X}, y_*^t > f_*} \gamma_{y_*^t}(\boldsymbol{x})$, and ρ_T is the maximum information gain of at most T selected points.

3.5. Model Adaptation

In practice we do not know the hyper-parameters of the GP, so we must adapt our GP model as we observe more data. A standard way to learn the GP hyper-parameters is to optimize the marginal data likelihood with respect to the hyper-parameters. As a full Bayesian treatment, we can also draw samples of the hyper-parameters using slice sampling (Vanhatalo et al., 2013), and then marginalize out the hyper-parameters in our acquisition function in Eq. (6). Namely, if we use E to denote the set of sampled settings for the GP hyper-parameters, our acquisition function becomes

$$\alpha_t(x) = \sum_{\eta\in E}\sum_{y_*\in Y_*}\left[\frac{\gamma_{y_*}^\eta(\boldsymbol{x})\psi(\gamma_{y_*}^\eta(\boldsymbol{x}))}{2\Psi(\gamma_{y_*}^\eta(\boldsymbol{x}))} - \log(\Psi(\gamma_{y_*}^\eta(\boldsymbol{x})))\right],$$

where $\gamma^{\eta}_{y_*}(\boldsymbol{x}) = \frac{y_* - \mu^{\eta}_t(\boldsymbol{x})}{\sigma^{\eta}_t(\boldsymbol{x})}$ and the posterior inference on the mean function μ^{η}_t and σ^{η}_t depends on the GP hyper-parameter setting η. Similar approaches have been used in (Hernández-Lobato et al., 2014; Snoek et al., 2012).

4. High Dimensional MES with Add-GP

The high-dimensional input setting has been a challenge for many BO methods. We extend MES to this setting via additive Gaussian processes (Add-GP). In the past, Add-GP has been used and analyzed for GP-UCB (Kandasamy et al., 2015), which assumed the high dimensional black-box function is a summation of several disjoint lower dimensional functions. Utilizing this special additive structure, we overcome the statistical problem of having insufficient data to recover a complex function, and the difficulty of optimizing acquisition functions in high dimensions.

Since the function components $f^{(m)}$ are independent, we can maximize the mutual information between the input in the active dimensions A_m and maximum of $f^{(m)}$ for each component separately. Hence, we have a separate acquisition function for each component, where $y^{(m)}$ is the evaluation of $f^{(m)}$:

$$\alpha^{(m)}_t(\boldsymbol{x}) = I(\{\boldsymbol{x}^{A_m}, y^{(m)}\}; y^{(m)}_* \mid D_t) \tag{8}$$

$$= H(p(y^{(m)} \mid D_t, \boldsymbol{x}^{A_m})) - \mathbb{E}[H(p(y^{(m)} \mid D_t, \boldsymbol{x}^{A_m}, y^{(m)}_*))] \tag{9}$$

$$\approx \sum_{y^{(m)}_*} \gamma^{(m)}_{y_*}(\boldsymbol{x}) \frac{\psi(\gamma^{(m)}_{y_*}(\boldsymbol{x}))}{2\Psi(\gamma^{(m)}_{y_*}(\boldsymbol{x}))} - \log(\Psi(\gamma^{(m)}_{y_*}(\boldsymbol{x}))) \tag{10}$$

where $\gamma^{(m)}_{y_*}(\boldsymbol{x}) = \frac{y^{(m)}_* - \mu^{(m)}_t(\boldsymbol{x})}{\sigma^{(m)}_t(\boldsymbol{x})}$. Analogously to the non-additive case, we sample $y^{(m)}_*$, separately for each function component. We select the final x_t by choosing a sub-vector $x^{(m)}_t \in \arg\max_{\boldsymbol{x}^{(m)} \in A_m} \alpha^{(m)}_t(\boldsymbol{x}^{(m)})$ and concatenating the components.

Sampling $y^{(m)}_*$ with a Gumbel distribution. The Gumbel sampling from Section 3.1 directly extends to sampling $y^{(m)}_*$, approximately. We simply need to sample from the component-wise CDF $\widehat{\Pr}[y^{(m)}_* < z] = \prod_{\boldsymbol{x} \in \hat{\mathfrak{X}}} \Psi(\gamma^{(m)}_y(\boldsymbol{x})))$, and use the same Gumbel approximation.

Sampling $y^{(m)}_*$ via posterior functions. The additive structure removes some connections on the input-to-hidden layer of our 1-hidden-layer neural network approximation $\tilde{f}(\boldsymbol{x}) = \boldsymbol{a}^{\mathrm{T}}_t \boldsymbol{\phi}(\boldsymbol{x})$. Namely, for each feature function ϕ there exists a unique group m such that ϕ is only active on $\boldsymbol{x}^{A_m}$, and $\phi(\boldsymbol{x}) = \sqrt{\frac{2}{D}} \cos(\omega^{\mathrm{T}} \boldsymbol{x}^{A_m} + c)$ where $\mathbb{R}^{|A_m|} \ni \omega \sim \hat{\kappa}^{(m)}(\omega)$ and $c \sim U[0, 2\pi]$. Similar to the non-additive case, we may draw a posterior sample $\boldsymbol{a}_t \sim \mathcal{N}(\boldsymbol{\nu}_t, \boldsymbol{\Sigma}_t)$ where $\boldsymbol{\nu}_t = \sigma^{-2} \boldsymbol{\Sigma}_t Z_t \boldsymbol{y}_t$ and $\boldsymbol{\Sigma}_t = (Z Z^{\mathrm{T}} \sigma^{-2} + \boldsymbol{I})^{-1}$. Let $B_m = \{i : \boldsymbol{\phi}_i(\boldsymbol{x}) \text{ is active on } \boldsymbol{x}^{A_m}\}$. The posterior sample for the function component $f^{(m)}$ is $\tilde{f}^{(m)}(\boldsymbol{x}) = (\boldsymbol{a}^{B_m}_t)^{\mathrm{T}} \phi^{B_m}(\boldsymbol{x}^{A_m})$. Then we can maximize $\tilde{f}^{(m)}$ to obtain a sample for $y^{(m)}_*$.

The algorithm for the additive max-value entropy search method (add-MES) is shown in Algorithm 2. The function APPROX-MI does the pre-computation for approximating the mutual information in a similar way as in Algorithm 1, except that it only acts on the active dimensions in the m-th group.

Algorithm 2 Additive Max-value Entropy Search

function Add-MES (f, D_0)
  for $t = 1, \cdots, T$ **do**
    for $m = 1, \cdots, M$ **do**
      $\alpha^{(m)}_{t-1}(\cdot) \leftarrow$ APPROX-MI (D_{t-1})
      $\boldsymbol{x}^{A_m}_t \leftarrow \arg\max_{\boldsymbol{x}^{A_m} \in \mathfrak{X}^{A_m}} \alpha^{(m)}_{t-1}(\boldsymbol{x})$
    end for
    $y_t \leftarrow f(\boldsymbol{x}_t) + \epsilon_t, \epsilon_t \sim \mathcal{N}(0, \sigma^2)$
    $\mathfrak{D}_t \leftarrow D_{t-1} \cup \{\boldsymbol{x}_t, y_t\}$
  end for
end function

5. Experiments

In this section, we probe the empirical performance of MES and add-MES on a variety of tasks. Here, MES-G denotes MES with y_* sampled from the approximate Gumbel distribution, and MES-R denotes MES with y_* computed by maximizing a sampled function represented by random features. Following (Hennig & Schuler, 2012; Hernández-Lobato et al., 2014), we adopt the zero mean function and non-isotropic squared exponential kernel as the prior for the GP. We compare to methods from the entropy search family, i.e., ES and PES, and to other popular Bayesian optimization methods including GP-UCB (denoted by UCB), PI, EI and EST. The parameter for GP-UCB was set according to Theorem 2 in (Srinivas et al., 2010); the parameter for PI was set to be the observation noise σ. For the functions with unknown GP hyper-parameters, every 10 iterations, we learn the GP hyper-parameters using the same approach as was used by PES (Hernández-Lobato et al., 2014). For the high dimensional tasks, we follow (Kandasamy et al., 2015) and sample the additive structure/GP parameters with the highest data likelihood when they are unknown. We evaluate performance according to the simple regret and inference regret as defined in Section 2.3. We used the open source Matlab implementation of PES, ES and EST (Hennig & Schuler, 2012; Hernández-Lobato

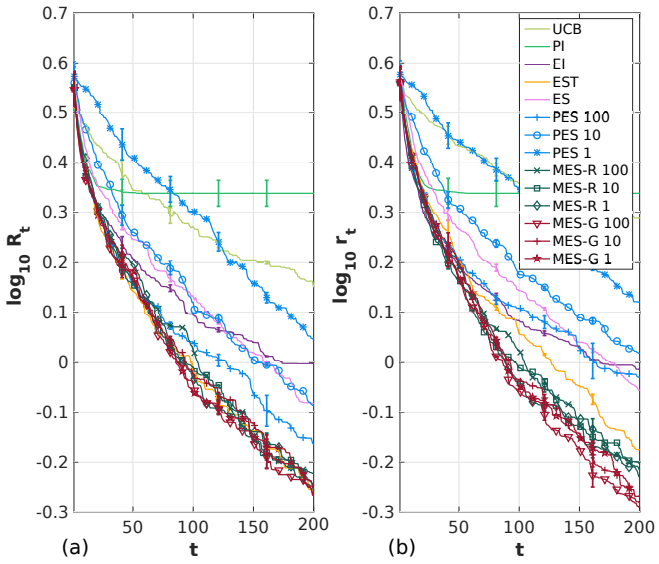

Figure 1. (a) Inference regret; (b) simple regret. MES methods are much less sensitive to the number of maxima y_* sampled for the acquisition function (1, 10 or 100) than PES is to the number of argmaxes x_*.

Table 1. The runtime of selecting the next input. PES 100 is significantly slower than other methods. MES-G's runtime is comparable to the fastest method EI while it performs better in terms of simple and inference regrets.

METHOD	TIME (S)	METHOD	TIME (S)
UCB	0.08 ± 0.05	PES 1	0.20 ± 0.06
PI	0.10 ± 0.02	MES-R 100	5.85 ± 0.86
EI	0.07 ± 0.03	MES-R 10	0.67 ± 0.11
EST	0.15 ± 0.02	MES-R 1	0.13 ± 0.03
ES	8.07 ± 3.02	MES-G 100	0.12 ± 0.02
PES 100	15.24 ± 4.44	MES-G 10	0.09 ± 0.02
PES 10	1.61 ± 0.50	MES-G 1	0.09 ± 0.03

et al., 2014; Wang et al., 2016). Our Matlab code and test functions are available at `https://github.com/zi-w/Max-value-Entropy-Search/`.

5.1. Synthetic Functions

We begin with a comparison on synthetic functions sampled from a 3-dimensional GP, to probe our conjecture that MES is much more robust to the number of y_* sampled to estimate the acquisition function than PES is to the number of x_* samples. For PES, we sample 100 (PES 100), 10 (PES 10) and 1 (PES 1) argmaxes for the acquisition function. Similarly, we sample 100, 10, 1 y_* values for MES-R and MES-G. We average the results on 100 functions sampled from the same Gaussian kernel with scale parameter 5.0 and bandwidth parameter 0.0625, and observation noise $\mathcal{N}(0, 0.01^2)$.

Figure 1 shows the simple and inference regrets. For both regret measures, PES is very sensitive to the the number of x_* sampled for the acquisition function: 100 samples lead to much better results than 10 or 1. In contrast, both MES-G and MES-R perform competitively even with 1 or 10 samples. Overall, MES-G is slightly better than MES-R, and both MES methods performed better than other ES methods. MES methods performed better than all other methods with respect to simple regret. For inference regret, MES methods performed similarly to EST, and much better than all other methods including PES and ES.

In Table 1, we show the runtime of selecting the next input per iteration[1] using GP-UCB, PI, EI, EST, ES, PES, MES-R and MES-G on the synthetic data with fixed GP hyper-parameters. For PES and MES-R, every x_* or y_* requires running an optimization sub-procedure, so their running time grows noticeably with the number of samples. MES-G avoids this optimization, and competes with the fastest methods EI and UCB.

In the following experiments, we set the number of x_* sampled for PES to be 200, and the number of y_* sampled for MES-R and MES-G to be 100 unless otherwise mentioned.

5.2. Optimization Test Functions

We test on three challenging optimization test functions: the 2-dimensional eggholder function, the 10-dimensional Shekel function and the 10-dimensional Michalewicz function. All of these functions have many local optima. We randomly sample 1000 points to learn a good GP hyper-parameter setting, and then run the BO methods with the same hyper-parameters. The first observation is the same for all methods. We repeat the experiments 10 times. The averaged simple regret is shown in the appendix, and the inference regret is shown in Table 2. On the 2-d eggholder function, PES was able to achieve better function values faster than all other methods, which verified the good performance of PES when sufficiently many x_* are sampled. However, for higher-dimensional test functions, the 10-d Shekel and 10-d Michalewicz function, MES methods performed much better than PES and ES, and MES-G performed better than all other methods.

5.3. Tuning Hyper-parameters for Neural Networks

Next, we experiment with Levenberg-Marquardt optimization for training a 1-hidden-layer neural network. The 4 parameters we tune with BO are the number of neurons, the damping factor μ, the μ-decrease factor, and the μ-increase factor. We test regression on the Boston housing dataset

[1]All the timing experiments were run exclusively on an Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz. The function evaluation time is excluded.

Table 2. Inference regret R_T for optimizing the eggholder function, Shekel function, and Michalewicz function.

METHOD	EGGHOLDER	SHEKEL	MICHALEWICZ
UCB	141.00 ± 70.96	9.40 ± 0.26	6.07 ± 0.53
PI	52.04 ± 39.03	6.64 ± 2.00	4.97 ± 0.39
EI	71.18 ± 59.18	6.63 ± 0.87	4.80 ± 0.60
EST	55.84 ± 24.85	5.57 ± 2.56	5.33 ± 0.46
ES	48.85 ± 29.11	6.43 ± 2.73	5.11 ± 0.73
PES	37.94 ± 26.05	8.73 ± 0.67	5.17 ± 0.74
MES-R	54.47 ± 37.71	6.17 ± 1.80	4.97 ± 0.59
MES-G	46.56 ± 27.05	5.45 ± 2.07	4.49 ± 0.51

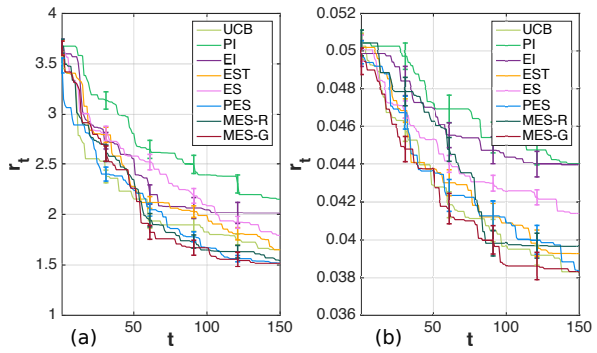

Figure 2. Tuning hyper-parameters for training a neural network, (a) Boston housing dataset; (b) breast cancer dataset. MES methods perform better than other methods on (a), while for (b), MES-G, UCB, PES perform similarly and better than others.

and classification on the breast cancer dataset (Bache & Lichman, 2013). The experiments are repeated 20 times, and the neural network's weight initialization and all other parameters are set to be the same to ensure a fair comparison. Both of the datasets were randomly split into train/validation/test sets. We initialize the observation set to have 10 random function evaluations which were set to be the same across all the methods. The averaged simple regret for the regression L2-loss on the validation set of the Boston housing dataset is shown in Fig. 2(a), and the classification accuracy on the validation set of the breast cancer dataset is shown in Fig. 2(b). For the classification problem on the breast cancer dataset, MES-G, PES and UCB achieved a similar simple regret. On the Boston housing dataset, MES methods achieved a lower simple regret. We also show the inference regrets for both datasets in Table 3.

5.4. Active Learning for Robot Pushing

We use BO to do active learning for the pre-image learning problem for pushing (Kaelbling & Lozano-Pérez, 2017). The function we optimize takes as input the pushing action of the robot, and outputs the distance of the pushed object to the goal location. We use BO to minimize the function in order to find a good pre-image for pushing the object to the designated goal location. The first function we tested has a 3-dimensional input: robot location (r_x, r_y) and pushing duration t_r. We initialize the observation size to be one, the same across all methods. The second function has a 4-dimensional input: robot location and angle (r_x, r_y, r_θ), and pushing duration t_r. We initialize the observation to be 50 random points and set them the same for all the methods. We select 20 random goal locations for each function to test if BO can learn where to push for these locations. We show the simple regret in Fig. 4 and the inference regret in Table 4. MES methods performed on a par with or better than their competitors.

Table 3. Inference regret R_T for tuning neural network hyper-parameters on the Boston housing and breast cancer datasets.

METHOD	BOSTON	CANCER (%)
UCB	1.64 ± 0.43	3.83 ± 0.01
PI	2.15 ± 0.99	4.40 ± 0.01
EI	1.99 ± 1.03	4.40 ± 0.01
EST	1.65 ± 0.57	3.93 ± 0.01
ES	1.79 ± 0.61	4.14 ± 0.00
PES	1.52 ± 0.32	3.84 ± 0.01
MES-R	1.54 ± 0.56	3.96 ± 0.01
MES-G	1.51 ± 0.61	3.83 ± 0.01

Table 4. Inference regret R_T for action selection in robot pushing.

METHOD	3-D ACTION	4-D ACTION
UCB	1.10 ± 0.66	0.56 ± 0.44
PI	2.03 ± 1.77	0.16 ± 0.20
EI	1.89 ± 1.87	0.30 ± 0.33
EST	0.70 ± 0.90	0.24 ± 0.17
ES	0.62 ± 0.59	0.25 ± 0.20
PES	0.81 ± 1.27	0.38 ± 0.38
MES-R	0.61 ± 1.23	0.16 ± 0.10
MES-G	0.61 ± 1.26	0.24 ± 0.25

5.5. High Dimensional BO with Add-MES

In this section, we test our add-MES algorithm on high dimensional black-box function optimization problems. First we compare add-MES and add-GP-UCB (Kandasamy et al., 2015) on a set of synthetic additive functions with known additive structure and GP hyper-parameters. Each function component of the synthetic additive function is active on at most three input dimensions, and is sampled from a GP with zero mean and Gaussian kernel (bandwidth = 0.1 and scale = 5). For the parameter of add-GP-UCB, we follow (Kandasamy et al., 2015) and set $\beta_t^{(m)} = |A_m| \log 2t/5$. We set the number of $y_*^{(m)}$ sampled for each function component in add-MES-R and add-MES-G to be 1. We repeat each experiment for 50 times

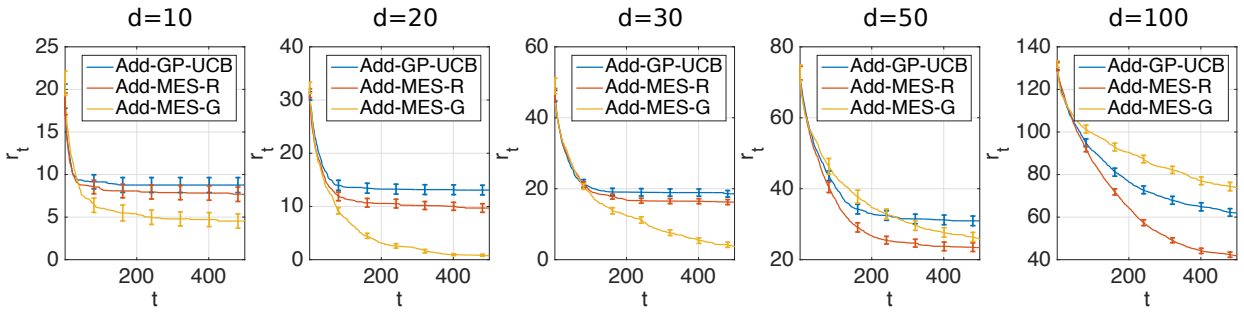

Figure 3. Simple regrets for add-GP-UCB and add-MES methods on the synthetic add-GP functions. Both add-MES methods outperform add-GP-UCB except for add-MES-G on the input dimension $d = 100$. Add-MES-G achieves the lowest simple regret when d is relatively low, while for higher d add-MES-R becomes better than add-MES-G.

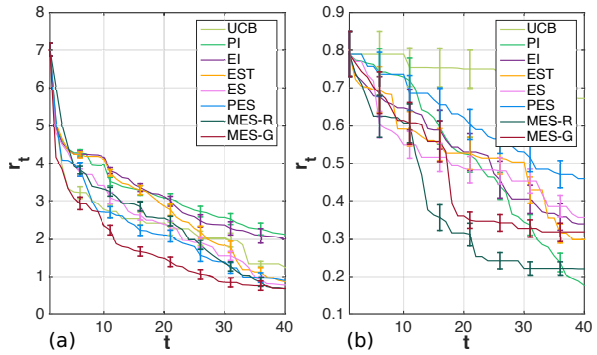

Figure 4. BO for active data selection on two robot pushing tasks for minimizing the distance to a random goal with (a) 3-D actions and (b) 4-D actions. MES methods perform better than other methods on the 3-D function. For the 4-D function, MES methods converge faster to a good regret, while PI achieves lower regret in the very end.

for each dimension setting. The results for simple regret are shown in Fig. 3. Add-MES methods perform much better than add-GP-UCB in terms of simple regret. Interestingly, add-MES-G works better in lower dimensional cases where $d = 10, 20, 30$, while add-MES-R outperforms both add-MES-G and add-GP-UCB for higher dimensions where $d = 50, 100$. In general, MES-G tends to overestimate the maximum of the function because of the independence assumption, and MES-R tends to underestimate the maximum of the function because of the imperfect global optimization of the posterior function samples. We conjecture that MES-R is better for settings where exploitation is preferred over exploration (e.g., not too many local optima), and MES-G works better if exploration is preferred.

To further verify the performance of add-MES in high dimensional problems, we test on two real-world high dimensional experiments. One is a function that returns the distance between a goal location and two objects being pushed by a robot which has 14 parameters[2]. The other function returns the walking speed of a planar bipedal robot, with 25 parameters to tune (Westervelt et al., 2007). In Fig. 5, we show the simple regrets achieved by add-GP-UCB and add-MES. Add-MES methods performed competitively compared to add-GP-UCB on both tasks.

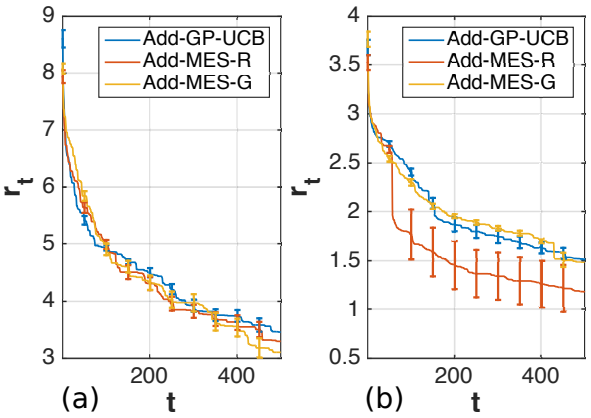

Figure 5. Simple regrets for add-GP-UCB and add-MES methods on (a) a robot pushing task with 14 parameters and (b) a planar bipedal walker optimization task with 25 parameters. Both MES methods perform competitively comparing to add-GP-UCB.

6. Conclusion

We proposed a new information-theoretic approach, max-value entropy search (MES), for optimizing expensive black-box functions. MES is competitive with or better than previous entropy search methods, but at a much lower computational cost. Via additive GPs, MES is adaptable to high-dimensional settings. We theoretically connected MES to other popular Bayesian optimization methods including entropy search, GP-UCB, PI, and EST, and showed a bound on the simple regret for a variant of MES. Empirically, MES performs well on a variety of tasks.

[2]We implemented the function in (Catto, 2011).

Acknowledgements

We thank Prof. Leslie Pack Kaelbling and Prof. Tomás Lozano-Pérez for discussions on active learning and Dr. William Huber for his solution to "Extreme Value Theory - Show: Normal to Gumbel" at `stats.stackexchange.com`, which leads to our Gumbel approximation in Section 3.1. We gratefully acknowledge support from NSF CAREER award 1553284, NSF grants 1420927 and 1523767, from ONR grant N00014-14-1-0486, and from ARO grant W911NF1410433. We thank MIT Supercloud and the Lincoln Laboratory Supercomputing Center for providing computational resources. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of our sponsors.

References

Auer, Peter. Using confidence bounds for exploitation-exploration tradeoffs. *Journal of Machine Learning Research*, 3:397–422, 2002.

Bache, Kevin and Lichman, Moshe. UCI machine learning repository. 2013.

Brochu, Eric, Cora, Vlad M, and De Freitas, Nando. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. Technical Report TR-2009-023, University of British Columbia, 2009.

Calandra, Roberto, Seyfarth, André, Peters, Jan, and Deisenroth, Marc Peter. An experimental comparison of Bayesian optimization for bipedal locomotion. In *International Conference on Robotics and Automation (ICRA)*, 2014.

Catto, Erin. Box2D, a 2D physics engine for games. `http://box2d.org`, 2011.

Duvenaud, David K, Nickisch, Hannes, and Rasmussen, Carl E. Additive Gaussian processes. In *Advances in Neural Information Processing Systems (NIPS)*, 2011.

Fisher, Ronald Aylmer. *The genetical theory of natural selection: a complete variorum edition*. Oxford University Press, 1930.

Hennig, Philipp and Schuler, Christian J. Entropy search for information-efficient global optimization. *Journal of Machine Learning Research*, 13:1809–1837, 2012.

Hernández-Lobato, José Miguel, Hoffman, Matthew W, and Ghahramani, Zoubin. Predictive entropy search for efficient global optimization of black-box functions. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

Hoffman, Matthew W and Ghahramani, Zoubin. Output-space predictive entropy search for flexible global optimization. In *NIPS workshop on Bayesian Optimization*, 2015.

Kaelbling, Leslie Pack and Lozano-Pérez, Tomás. Learning composable models of primitive actions. In *International Conference on Robotics and Automation (ICRA)*, 2017.

Kandasamy, Kirthevasan, Schneider, Jeff, and Poczos, Barnabas. High dimensional Bayesian optimisation and bandits via additive models. In *International Conference on Machine Learning (ICML)*, 2015.

Krause, Andreas and Ong, Cheng S. Contextual Gaussian process bandit optimization. In *Advances in Neural Information Processing Systems (NIPS)*, 2011.

Kushner, Harold J. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Fluids Engineering*, 86(1):97–106, 1964.

Lizotte, Daniel J, Wang, Tao, Bowling, Michael H, and Schuurmans, Dale. Automatic gait optimization with Gaussian process regression. In *International Conference on Artificial Intelligence (IJCAI)*, 2007.

Močkus, J. On Bayesian methods for seeking the extremum. In *Optimization Techniques IFIP Technical Conference*, 1974.

Neal, R.M. *Bayesian Learning for Neural networks*. Lecture Notes in Statistics 118. Springer, 1996.

Rahimi, Ali, Recht, Benjamin, et al. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems (NIPS)*, 2007.

Rasmussen, Carl Edward and Williams, Christopher KI. Gaussian processes for machine learning. *The MIT Press*, 2006.

Rudin, Walter. *Fourier analysis on groups*. John Wiley & Sons, 2011.

Snoek, Jasper, Larochelle, Hugo, and Adams, Ryan P. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems (NIPS)*, 2012.

Srinivas, Niranjan, Krause, Andreas, Kakade, Sham M, and Seeger, Matthias. Gaussian process optimization in the bandit setting: no regret and experimental design. In *International Conference on Machine Learning (ICML)*, 2010.

Thornton, Chris, Hutter, Frank, Hoos, Holger H, and Leyton-Brown, Kevin. Auto-WEKA: combined selection and hyperparameter optimization of classification algorithms. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2013.

Vanhatalo, Jarno, Riihimäki, Jaakko, Hartikainen, Jouni, Jylänki, Pasi, Tolvanen, Ville, and Vehtari, Aki. Gpstuff: Bayesian modeling with gaussian processes. *Journal of Machine Learning Research*, 14(Apr):1175–1179, 2013.

Wang, Zi, Zhou, Bolei, and Jegelka, Stefanie. Optimization as estimation with Gaussian processes in bandit settings. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2016.

Wang, Zi, Jegelka, Stefanie, Kaelbling, Leslie Pack, and Lozano-Pérez, Tomás. Focused model-learning and planning for non-Gaussian continuous state-action systems. In *International Conference on Robotics and Automation (ICRA)*, 2017.

Westervelt, Eric R, Grizzle, Jessy W, Chevallereau, Christine, Choi, Jun Ho, and Morris, Benjamin. *Feedback control of dynamic bipedal robot locomotion*, volume 28. CRC press, 2007.