WARWICK
THE UNIVERSITY OF WARWICK

Original citation:
Pearce, Michael and Branke, Juergen (2017) Efficient information collection on portfolios. Working Paper.

Permanent WRAP URL:
http://wrap.warwick.ac.uk/85566

Copyright and reuse:
The Warwick Research Archive Portal (WRAP) makes this work by researchers of the University of Warwick available open access under the following conditions. Copyright © and all moral rights to the version of the paper presented here belong to the individual author(s) and/or other copyright owners. To the extent reasonable and practicable the material made available in WRAP has been checked for eligibility before being made available.

Copies of full items can be used for personal research or study, educational, or not-for-profit purposes without prior permission or charge. Provided that the authors, title and full bibliographic details are credited, a hyperlink and/or URL is given for the original metadata page and the content is not changed in any way.

A note on versions:
The version presented here is a working paper or pre-print that may be later published elsewhere. If a published version is known of, the above WRAP URL will contain details on finding it.

For more information, please contact the WRAP Team at: wrap@warwick.ac.uk

warwick.ac.uk/lib-publications

Efficient Information Collection on Portfolios

Michael Pearce[a], Juergen Branke[b]

[a] *Complexity Science, The University of Warwick, Coventry, CV4 7AL, United Kingdom, m.a.l.pearce@warwick.ac.uk*

[b] *Warwick Business School, The University of Warwick, Coventry, CV4 7AL, United Kingdom, Juergen.Branke@wbs.ac.uk*

Abstract

This paper tackles the problem of efficiently collecting data to learn a classifier, or mapping, from each task to the best performing tool, where tasks are described by continuous features and there is a portfolio of tools to choose from. A typical example is selecting an optimization algorithm from a portfolio of algorithms, based on some features of the problem instance to be solved. Information is collected by testing a tool on a task and observing its (possibly stochastic) performance. The goal is to minimize the opportunity cost of the constructed mapping, where opportunity cost is the difference between the performance of the true best tool for each task, and the performance of the tool chosen by the constructed mapping, summed over all tasks. We propose several fully sequential information collection policies based on Bayesian statistics and Gaussian Process models. In each step, they myopically sample the (task, tool) pair that promises the highest value of the information collected. We prove optimality under certain conditions and empirically demonstrate that our methods significantly outperform standard approaches on a set of synthetic benchmark problems.

Keywords: Global Optimization, Information Collection, Optimal Learning, Gaussian Processes, Algorithm Selection

1. Introduction

We consider the problem of sampling to efficiently identify a mapping from a finite set of tasks to the best tool for each task from a portfolio of tools. Given a budget of measurements to sample the performance of one tool applied to one task, the goal is to identify a mapping that minimizes the expected regret, or opportunity cost, the difference in expected performance of the true best tool and the tool selected by the mapping, summed over all tasks. We approach the problem as a variant of ranking and selection problems, where an experimenter is typically required to find the single best overall tool from a portfolio where best is defined as having the best expected performance which can only be inferred via sampling. The main difference of our problem compared to the typical ranking and selection problem is that we aim to simultaneously identify the best tool for each task, where tasks can be described by features in $\mathbb{R}^D$, and there is some correlation of tool performance across tasks with similar features.

This problem has many practical applications, including the following three examples.

1. *Algorithm portfolios.* For most hard optimization problems, different algorithms have been developed. Although some algorithms may work better than others overall, usually different algorithms work best for different problem instances. Thus, there is the problem of deciding which algorithm to use for which problem instance, based on features of the problem instance. Smith-Miles (2008) provides a survey on this algorithm selection problem. To learn about the mapping, we can sample any particular algorithm on any particular problem instance, and will observe a (possibly noisy) performance value as a result. Be-

cause the potential set of problem instances is large and running an algorithm on a problem instance is computationally expensive, it is important to collect information efficiently, and make intelligent decisions on which (problem instance, algorithm)-pairs to sample to obtain the best possible mapping.

2. *Personalized Medicine.* The pharmaceutical industry is currently experiencing a shift from the one-drug-fits-all paradigm towards personalization, where therapies are targeted towards particular groups of patients. Clinical trials then not only have to determine whether a drug is effective or not, but also which drug works best for which type of patient, depending on patient characteristics. Our algorithm could be used to reduce the number of clinical experiments needed to derive this mapping from patient characteristics to drug, or find a better mapping for a given number of experiments. A similar problem has been considered in Xu et al. (2014).

3. *Online marketing.* In online advertisement, it is easy to deploy several different advertisements and advertisement formats (banner, video, etc.) simultaneously, and pick for each viewer the advertisement that one believes results in the highest return (in terms of click-through rate or money spent). Often, some information is available on the viewers, such as search terms, websites visited or order history. If we appropriately define the feature space to describe the viewers, our algorithms could be used to efficiently learn which advertisement would be most effective based on some viewer characteristics.

We tackle the problem of information collection on a portfolio by using Gaussian Processes as a metamodel to predict a tool's expected performance on all (measured and unmeasured) tasks, and myopic sampling policies that try to maximize the value of information of the next sample taken. We propose and empirically compare three such policies, REVI, NEVI and EVI. For a given budget, our three policies sequentially create sample designs producing mappings that perform significantly better than Latin Hypercube designs reducing the necessary sampling budget to obtain a desired level of performance by up to 67% in our experiments. Also, they significantly outperform the single best tool for all tasks in our synthetic benchmarks.

The paper is structured as follows. Section 2 provides a brief overview of related work. In Section 3 the problem and mathematical framework are laid out, followed in Section 4 by the derivation of our methods. Empirical results are reported in Section 5. We conclude in Section 6 with a summary and some ideas for future work.

2. Related Work

The problem of identifying the best tool for one single task is a typical ranking and selection problem, and we adapt sequential sampling techniques from ranking and selection to our problem of information collection on a portfolio. Branke et al. (2007) provide a review and comparison of several ranking and selection methods, a more recent survey focusing on algorithms derived under the Bayesian framework has been written by Chen et al. (2015). Myopic sampling policies sequentially sample from alternatives in such a way that at each step, the sample has the highest expectation of improving the objective, which typically is either probability of correct selection or expected opportunity cost. The methods we propose are myopic, and follow the framework of the Knowledge Gradient method investigated in Frazier et al. (2008) and Chick et al. (2010) but going back to Gupta and Miescke (1996). Frazier et al. (2009) extend the method to the case of correlated alternatives, i.e., when sampling from one alternative may teach us something about the performance of similar alternatives.

In the case when the search space is continuous, Gaussian Processes are often used as statistical model of the objective function given the data collected. In a deterministic setting, the popular Efficient Global Optimization (EGO) algorithm (Jones et al., 1998) uses a Gaussian Process model combined with a simple expected improvement criterion to sequentially decide which point in the search space to evaluate next to efficiently find the global optimum. The Sequential Kriging Optimization algorithm (Huang et al., 2006) extends the EGO algorithm to the case when function evaluations are noisy. Scott et al. (2011) extend the discrete Knowledge Gradient policy to the continuous case which is myopically optimal under certain conditions.

Finding contours or level sets of a function over an input domain can be viewed as finding the best of two functions by finding the zero level set of the difference between the two functions. Picheny et al. (2010) use a Gaussian Prcoesses model to approximate a function and propose a sequential sampling procedure to accurately approximate the function within a target region defined by a level set. Bingham et al. (2014) provide an overview of expected improvement methods with an example for contour estimation for volcano data.

In our paper, we use Gaussian Processes to model the performance of a tool on the set of tasks as described by their features, and we propose a myopically optimal sampling policy to efficiently identify the best tool for each task, or the highest function for each point in a finite domain. The aim is to minimize the expected opportunity cost of the learned mapping, which is the difference in performance between the selected alternative and the true best alternative, summed over all tasks.

The problem of learning which drug is most effective for which patient has already been considered by Xu et al. (2014). In this work, patients are characterized by biomarkers and treatment response is binary, and the approach proposed is rather heuristic and does not have any optimality guarantees.

In the algorithm selection literature (Rice, 1976; Smith-Miles, 2008), quite a few researchers have tried to understand which algorithm works best on which problem. However, they usually focus on the identification of features suitable for classification, or the design of classification algorithms, while assuming that the data on the performance of a variety of algorithms on a large number of test problems is given. To the best of our knowledge, no one so far has looked at efficiently collecting data in order to learn a mapping from problem instance to algorithm that maximizes performance.

3. Problem Definition

We assume we are given a finite set of M tasks indexed by $i \in \{1, ..., M\}$ and a set of A alternative tools indexed by $a \in \{1, ..., A\}$. Each task i can be characterized by D continuous features, $x_i \in \mathbb{R}^D$, and the set of feature vectors for all tasks is denoted $X = \{x_1, ..., x_M\}$. We can apply a tool a to a task i to obtain a stochastic performance measurement that is a realization of a random variable $Y_{i,a} = \zeta_{i,a} + \epsilon_a$ where $\zeta_a \in \mathbb{R}^M$ is the unknown vector of M expected performance values for all tasks for tool a and $\epsilon_a \sim N(0, \sigma^2_{\epsilon,a})$ is white noise distributed with known variance which in practice can be estimated. The values of $\zeta_1, ..., \zeta_A$ are assumed to come from underlying deterministic latent functions of the task features $\tilde{\zeta}_a(x) : \mathbb{R}^D \to \mathbb{R}$ and $\zeta_{i,a} = \tilde{\zeta}_a(x_i)$. Given a finite budget of N performance measurements, or samples, to be allocated to the (task, tool)-design space $\{1, ..., M\} \times \{1, ..., A\}$, the goal is to find a classifier, or a mapping, $S : X \to \{1, ..., A\}$ from features to the best tool such that $S(x_i)$ approximates $\underset{a}{\operatorname{argmax}}\, \zeta_{i,a}$ for each $x_i \in X$ and therefore maximizes the expected portfolio performance summed over all considered tasks:

$$\text{Portfolio Performance} = \sum_{i=1}^{M} w_i \zeta_{i,S(x_i)} \tag{1}$$

where w_i are known positive weights representing the relative importance of each task. This objective is equivalent to minimizing the opportunity cost (OC)

$$\text{OC} = \sum_{i=1}^{M} w_i \left(\max_a \zeta_{i,a} - \zeta_{i,S(x_i)} \right), \tag{2}$$

the weighted performance difference between the true best tool and the chosen tool summed over all considered tasks. The objective of sampling is the expected portfolio performance and not the classification error and so we refer to $S(x)$ as a mapping, as opposed to a classifier.

Performance measurements, or samples, may be allocated sequentially so that after observing n samples the experimenter may choose which tool and task to sample for the $(n+1)^{th}$ sample until the budget is exhausted. If $M = 1$ then the problem reduces to an uncorrelated ranking and selection problem.

4. Methodologies

We start this section by introducing a mathematical framework followed by three purely myopic (stationary, Markovian, deterministic) sampling policies in order of decreasing complexity but also decreasing efficiency in terms of improving the mapping.

For each tool we treat the unknown true performance values, ζ_a, as Bayesian random variables denoted by θ_a. Given a multivariate normal prior $\theta_a \sim \mathcal{N}(\mu_a^0, \Sigma_a^0)$ and Gaussian observation noise, the likelihood is also Gaussian and therefore the posterior after n samples is also multivariate Gaussian $\theta_a \sim \mathcal{N}(\mu_a^n, \Sigma_a^n)$. We assume that the performance of a tool on two tasks is correlated, and that the covariance can be modeled by a kernel function of the task features $(\Sigma_a)_{i,j} = k(x_i, x_j)$. Thus, essentially this is a Gaussian process regression model, but it is discretized by only evaluating at points $x \in X$. Explicit formulae are given below and further details about Gaussian Process Regression can be found in Rasmussen and Williams (2004) and Sacks et al. (2012).

Consider a state during sampling after which n samples have been collected. We denote the sequence of sampled tasks $(i^1, ..., i^n)$, tools $(a^1, ..., a^n)$ and the sequence of pairs $(i, a)^1, ..., (i, a)^n$ is written as $\{(i, a)\}_1^n$. The vector of corresponding performance measurements is denoted $(y^1, ..., y^n) = Y^n$ and the subset of performance measurements from tool a is $Y_a^n \subseteq Y^n$. We next define the filtration, $\mathcal{F}^n$, to be the sigma algebra generated by the tasks, tools and performance measurements sampled so far $\mathcal{F}^n = \sigma\{(i^1, a^1, y^1), ..., (i^n, a^n, y^n)\}$. We next define the sequence of feature values of sampled tasks $(x_{i^1}, x_{i^2}, ..., x_{i^n}) = W^n$ and denote the sub-sequences $W_a^n \subseteq W^n$ that contain only elements relating to samples from tool a. By using the Matrix Inversion Lemma (Hager, 1989) to condition the prior on the data points collected so far, the posterior mean and covariance for a single tool a, $\mathbb{P}[\theta_a|\mathcal{F}^n] = \mathcal{N}(\mu_a^n, \Sigma_a^n)$, are then given by the following matrix equations:

$$\mu_a^n = \mu_a^0 + (k_a^n)^\intercal (K_a^n)^{-1} Y_a^n, \tag{3}$$

$$\Sigma_a^n = \Sigma_a^0 - (k_a^n)^\intercal (K_a^n)^{-1} k_a^n, \tag{4}$$

where $k_{ij}^n = k(x_i, w_j)$ is the kernel evaluated between the i^{th} element in X and the j^{th} element in W_a^n. $K_{ij,a}^n = k(w_i, w_j) + \mathbb{1}_{\{i=j\}}\sigma_{\epsilon,a}^2$ is the matrix composed of the kernel evaluated between all possible pairs of tasks in W_a^n with added variance on the diagonal entries that account for noise.

True exact vectors of $\zeta_1, ..., \zeta_A$ are unknown to the experimenter, therefore the mapping is constructed by selecting the tool for each task with the highest predicted performance,

$$S^n(x_i) = \operatorname*{argmax}_a \mu_{i,a}^n,$$

and the predicted portfolio performance at time n is given by

$$P^n = \sum_{i=1}^{M} w_i \mu_{i,S(x_i)}^n = \sum_{i=1}^{M} w_i \max_a \mu_{i,a}^n \tag{5}$$

which we want to maximize.

At a given stage n during sampling, measuring the performance of tool a^{n+1} applied to task i^{n+1} generates the next performance value y^{n+1}. By concatenating the appropriate values to form the updated matrices $k_{a^{n+1}}^{n+1}, K_{a^{n+1}}^{n+1}$ and $Y_{a^{n+1}}^{n+1}$, one can use the Matrix Inversion Lemma again for the updated inverse $(K_{a^{n+1}}^{n+1})^{-1}$ to find the following recursion (which is also found in Frazier et al. (2009)) and does not contain the $(K_a^n)^{-1}$ matrix inversion, i^{n+1}

has temporarily been replaced by i for brevity:

$$\mu_a^{n+1} = \begin{cases} \mu_a^n + \frac{y^{n+1} - \mu_{i,a}^n}{\Sigma_{ii,a}^n + \sigma_{\epsilon,a}^2} \Sigma_{i,a}^n & a = a^{n+1} \\ \mu_a^n & otherwise \end{cases} \tag{6}$$

$$\Sigma_a^{n+1} = \begin{cases} \Sigma_a^n - \frac{\Sigma_{i,a}^n \Sigma_{i,a}^{n\intercal}}{\Sigma_{ii,a}^n + \sigma_{\epsilon,a}^2} & a = a^{n+1} \\ \Sigma_a^n & otherwise \end{cases} \tag{7}$$

where $\Sigma_{i,a}^n$ denotes the i^{th} column of the symmetric matrix Σ_a^n. Note that the Σ_a^n matrices only depend on the sampling decisions $\{(a,i)\}_1^n$. At time n, the scalar y^{n+1} and vector $\mu_{a^{n+1}}^{n+1}$ are not $\mathcal{F}^n$ measurable. However given the next (task, tool) pair to be sampled is $(i,a)^{n+1}$, and the prior predictive distribution of y^{n+1} conditioned on $\mathcal{F}^n$, $y^{n+1} \sim N(\theta_{i,a}, \sigma_{\epsilon,a}^2) = N(\mu_i^n, \Sigma_{ii,a}^n + \sigma_{a,\epsilon}^2)$, the distribution of $\mu_{a^{n+1}}^{n+1}$ conditioned on $\mathcal{F}^n$ can be calculated using the above recursion formula. For a given standard normal random variable $Z \sim N(0,1)$ we have the following (for clarity we have dropped the subscript a for the following two formulae):

$$\begin{aligned} \mu^{n+1} &= \mu^n + \frac{(\mu_i^n + \sqrt{\Sigma_{ii}^n + \sigma_\epsilon^2} Z) - \mu_i^n}{\Sigma_{ii}^n + \sigma_\epsilon^2} \Sigma_i^n \\ &= \mu^n + \frac{\Sigma_i^n}{\sqrt{\Sigma_{ii}^n + \sigma_\epsilon^2}} Z. \end{aligned}$$

And so we define the vector valued function $\tilde{\sigma}^n : \{1,..,M\} \times \{1,..,A\} \to \mathbb{R}^M$ with entries

$$\tilde{\sigma}^n(i,a) = \frac{\Sigma_{i,a}^n}{\sqrt{\Sigma_{ii,a}^n + \sigma_{\epsilon,a}^2}} \tag{8}$$

and therefore when conditioned on $\mathcal{F}^n$ we have

$$\begin{aligned} \mu_a^{n+1} &\sim \mathcal{N}(\mu_a^n, \tilde{\sigma}^n(i,a)\tilde{\sigma}^n(i,a)^\intercal), \\ \Sigma_a^{n+1} &= \Sigma_a^n - \tilde{\sigma}^n(i,a)\tilde{\sigma}^n(i,a)^\intercal, \end{aligned}$$

where the marginal distributions are given by $\mu_{j,a}^{n+1} \sim N(\mu_{j,a}^n, (\tilde{\sigma}_j^n(i,a))^2)$ for all j given the next sample will be at task tool (i,a). With a prior predictive distribution for the posterior mean after a new sample, we can calculate the expectation of predicted portfolio performance after the next sample. We now use this to define three sampling policies.

4.1. Regional Expected Value of Improvement Policy, REVI

We define the Regional Expected Value of Improvement of a new sample at task i and tool a as the expectation of improvement in predicted portfolio performance:

$$\mathrm{REVI}^n(i,a) = \mathbb{E}\left[P^{n+1}\middle|\mathcal{F}^n, (i,a)^{n+1} = (i,a)\right] - P^n, \tag{9}$$

and the formula may be computed analytically:

$$\mathrm{REVI}^n(i,a) = \mathbb{E}\left[\sum_{j=1}^{M} w_j \max_b \mu_{j,b}^{n+1}\middle|\mathcal{F}^n, (i,a)^{n+1} = (i,a)\right] - \sum_{j=1}^{M} w_j \max_b \mu_{j,b}^n \tag{10}$$

$$= \sum_j w_j \mathbb{E}\left[\max\{0, -|\mu_{j,a}^n - \max_{b\neq a} \mu_{j,b}^n| + \tilde{\sigma}_j^n(i,a)Z\}\right] \tag{11}$$

$$= \sum_j w_j \mathrm{h}(\Delta_{j,a}^n, \tilde{\sigma}_j^n(i,a)), \tag{12}$$

where the intermediate steps between Equations 10 and 11 are provided in the online appendix, $\Delta_{j,a}^n = |\mu_{j,a}^n - \max_{b\neq a} \mu_{j,b}^n|$ and the function $\mathrm{h} : \mathbb{R}\times\mathbb{R} \to \mathbb{R}$ is the well known expected improvement function of a normal random variable found by integrating over the truncated normal distribution of Z:

$$\mathrm{h}(\Delta, \sigma) = |\sigma|\phi(\Delta/|\sigma|) - \Delta\Phi(-\Delta/|\sigma|), \tag{13}$$

where ϕ and Φ are standard normal density and distribution functions, respectively. In this case $|\sigma|$ is necessary because $\tilde{\sigma}_j^n(i,a)$ may be negative and only the magnitude is necessary by the symmetry of the normal distribution. At a given stage during sampling, under the REVI policy, the next sample is allocated to the (task, tool) pair that satisfy:

$$(i,a)^{n+1} = \text{argmax REVI}^n(i,a). \tag{14}$$

The REVI sampling policy allocates each sample to maximize the expected improvement in the predicted portfolio performance and thus is myopically optimal by construction. It is also asymptomatically optimal, meaning that given an infinite sampling budget, the policy will always find the true best tool for each task and find the mapping that maximizes the true portfolio performance. This is because the expected improvement of sampling a (task, tool) pair decreases, on average, towards zero as more samples are allocated and thus any unsampled (task, tool) pair eventually becomes the (task, tool) pair that maximizes expected improvement and is chosen for sampling. Therefore as the budget approaches infinity, all (task,tool) pairs are sampled infinitely often and posterior distributions $\theta_1, ..., \theta_A$ become point masses at the true values $\zeta_1, ..., \zeta_A$:

Theorem 4.1. *When sampling according to the REVI policy* $\lim_{N\to\infty} S^N(x_i) = \underset{a}{\text{argmax}}\ \zeta_{i,a}$ *for all i.*

A more formal proof of Theorem 4.1 can be found in the online appendix. We also provide a Dynamic Programming formulation for this problem and give a bound on the sub optimality gap between the value of an optimal policy and the REVI policy for finite budgets.

REVI allocates samples based on a trade off between three considerations. Ceteris paribus, priority is given to (task, tool) pairs which have large posterior variance, low difference in posterior means between the selected tool and best of the other tools, and tasks whose performance is highly correlated to many other tasks. When using the squared exponential kernel for the Gaussian process, highly correlated tasks have similar features hence the REVI policy gives sampling priority to tasks in task-dense regions of feature space whose results may greatly influence the mapping and deprioritizes sampling of tasks with outlying features. It is for this reason we give this policy the name of Regional Expected Value of Improvement. Figure 1 provides an example comprising two tools and 50 tasks with features in $\mathbb{R}$ and Gaussian Processes with the squared exponential kernel. One can see that the REVI function is larger where tasks are dense and where there is large uncertainty about which tool is the best, i.e., where posterior means are close and variance is large.

Complete computation of $\mathrm{REVI}^n(i,a)$ for all (i,a) requires M^2A function calls to $\mathrm{h}(\Delta,\sigma)$ and also requires the entire matrices $\Sigma_1^n,...,\Sigma_A^n$ and all evaluations of $\tilde{\sigma}^n(i,a)$ which each have an M^2A memory requirement. This can be prohibitively expensive in scenarios with many tasks where M is large. The following two policies make simplifying assumptions that reduce this computational complexity.

4.2. Noisy Expected Value of Improvement Policy, NEVI

The NEVI policy assumes that the (task, tool) pair that maximizes the expected improvement in a tool's predicted performance on the selected task also maximizes the expected improvement in the portfolio predicted performance. It is therefore possible to approximate the sum in Equation 10 by taking only the i^{th} term reducing the computational complexity to $O(MA)$. Intuitively, the NEVI policy neglects the impact the sample would have on the posterior performance distribution of other correlated tasks.

We define the Noisy Expected Value of Improvement of a new sample at task i and tool a as the expected improvement in tool performance for the sampled task alone:

$$\begin{aligned} \mathrm{NEVI}^n(i,a) &= w_i \mathbb{E}\left[\max_b \mu_{i,b}^{n+1} \middle| \mathcal{F}^n, a^{n+1}=a\right] - \max_b w_i \mu_{i,b}^n & (15) \\ &= w_i \mathrm{h}(\Delta_{i,a}^n, \tilde{\sigma}_i^n(i,a)), & (16) \end{aligned}$$

and the next sample is determined by maximizing the above improvement:

$$(i,a)^{n+1} = \operatorname{argmax} \mathrm{NEVI}^n(i,a). \qquad (17)$$

The NEVI policy allocates samples to (task, tool) pairs based on a trade off between only two considerations, where the posterior means of the sampled tool and the best of the other tools is close, and where the posterior variance is large for the sampled (task, tool) pair. This policy does not account for the effect a new measurement will have on covarying predictions but it does account for noisy measurements which is why it is called the Noisy Expected Value of Improvement policy.

The NEVI policy is not myopically optimal but like the REVI policy, it is asymptotically optimal. We show in Section 3 that the NEVI and REVI policies perform comparably in our synthetic benchmarks when task features are uniformly distributed, whereas REVI outperforms NEVI when task features are clustered. Figure 1 shows how NEVI and REVI differ, for example the NEVI function gives more weight than REVI to outlying tasks. In the special case where there is no covariance between tasks, the NEVI and REVI policies allocate samples equally and therefore NEVI is also myopically optimal. In the special case where there is only one task, the REVI and NEVI policies become identical and both are equivalent to the Knowledge Gradient policy for sampling from A uncorrelated alternatives.

Each iteration of the NEVI policy only requires MA function calls to $\mathrm{h}(\Delta, \sigma)$ and only the diagonal elements of the $\Sigma_1^n, ..., \Sigma_A^n$ and single values $\tilde{\sigma}_i(i, a)$ which in total have a memory requirement that scales as MA. Thus the NEVI policy is much more efficient to compute than REVI for large M. However, one cannot use the recursion formula given in Equations 6 and 7. Instead the typically smaller $(K_a^n)^{-1}$ matrix inversion is required in Equations 3 and 4 and the computational complexity can be reduced by using formula for inverse of partitioned matrices given in Press et al. (1996) p. 77.

4.3. Expected Value of Improvement Policy, EVI

In addition to the simplifying assumption of NEVI, the Expected Value of Improvement (EVI) policy also assumes that the noise in performance measurements is negligible for the (task, tool) pair that maximizes the NEVI function. Therefore $\tilde{\sigma}_i^n(i, a) = \Sigma_{ii,a}^n / \sqrt{\Sigma_{ii,a}^n + \sigma_{\epsilon,a}} \approx \sqrt{\Sigma_{ii,a}^n}$ and $\mu_{i,a}^{n+1}$ is equal in distribution to $\theta_{i,a}$. We define the EVI of a new sample at task i and tool a as the following:

$$\begin{aligned} \mathrm{EVI}^n(i, a) &= w_i \mathbb{E}\left[\max\{\theta_{i,a}, \max_{b \neq a} \mu_{i,b}^n\} \middle| \mathcal{F}^n\right] - \max_b \mu_{i,b}^n && (18) \\ &= w_i \mathrm{h}\left(\Delta_{i,a}^n, \sqrt{\Sigma_{ii,a}^n}\right), && (19) \end{aligned}$$

and the next sample is given by maximising the above expected improvement:

$$(i, a)^{n+1} = \mathrm{argmax}\ \mathrm{EVI}^n(i, a). \qquad (20)$$

We include this policy for it's simplicity and we demonstrate numerically that it performs similarly to the REVI and NEVI policies when tasks are uniformly distributed and sampling budgets are small. However, like the

NEVI policy it loses some efficiency when tasks are clustered. When the variance of the white noises for each tool $\sigma^2_{\epsilon,1}, ..., \sigma^2_{\epsilon,A}$ are comparable to the posterior variances for each tool, the simplifying assumption of EVI become less applicable and EVI is less efficient. As sampling budget N increases, posterior variances for all the task and tools tends to zero therefore the EVI policy will always perform worse than the REVI and NEVI policies as budget increases.

In the example in Figure 1, NEVI and EVI are relatively similar and both have peaks for the same (task, tool) pair.

When performance measurements are deterministic, $\sigma_{\epsilon,a} = 0$ for all a, the EVI and NEVI policies allocate samples identically. The EVI policy is asymptotically optimal. This policy requires MA function calls to h(Δ, σ) and the diagonal elements of the posterior covariance matrices. It does not require computation or storage of $\tilde{\sigma}^n(i, a)$.

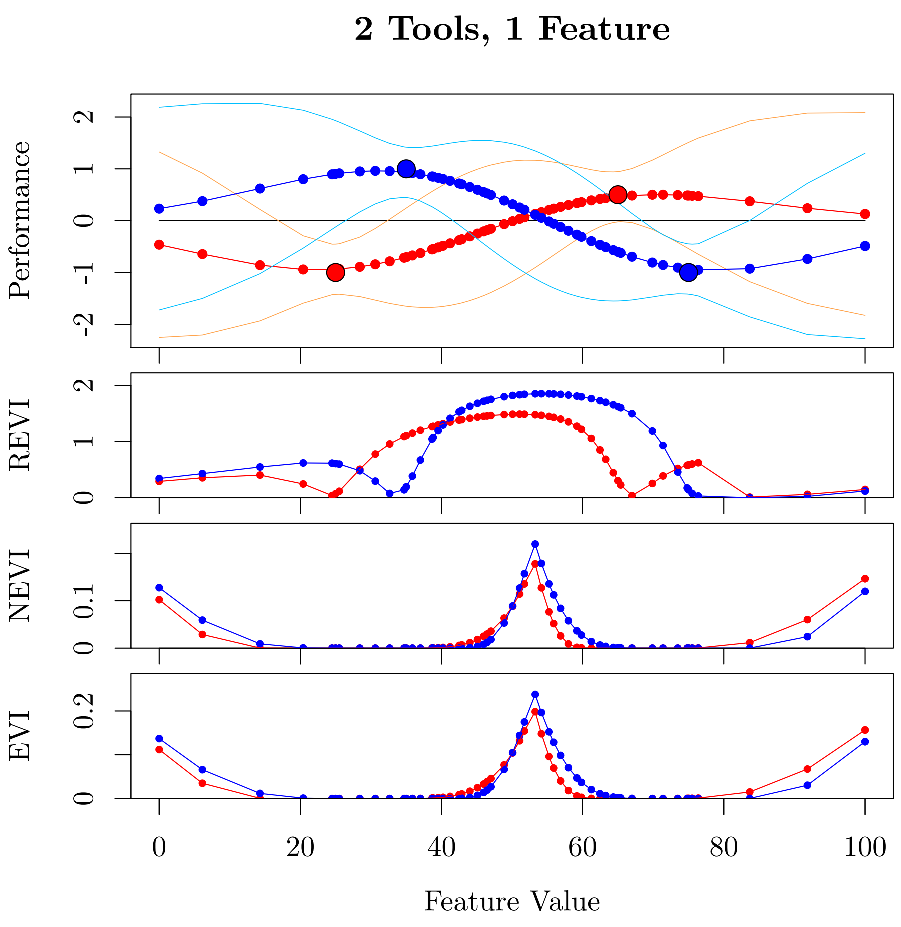

Figure 1: In all plots the x-axis is the single feature of the 50 tasks, $x_i \in \mathbb{R}$. Top: the Gaussian Processes for two tools after 4 performance measurements (large points) $(x_{i^1}, y^1), ..., (x_{i^4}, y^4)$ coloured according to tool, and posterior mean performance for 50 tasks (small points) $(x_i, \mu^4_{i,a})$ with confidence intervals. The aim of sampling is to maximise the highest means. Below: $\text{REVI}^4(i,a)$, $\text{NEVI}^4(i,a)$ and $\text{EVI}^4(i,a)$ plotted against x_i for both tools where all tasks have equal weight. REVI is high where task density is high, posterior means are close and posterior variance is large. NEVI and EVI don't account for task density therefore give relatively larger value to the outlying tasks.

5. Numerical Experiments

5.1. Experimental Setup

We create two artificial data sets of $M = 500$ tasks with all equal weights, $w_i = 1$ for all $i \in \{1, ..., 500\}$. The first set, the uniform case X_U, has feature values in the unit square $x_i \in (0,1)^2$ that are randomly uniformly distributed. The second set of tasks, the bimodal case X_B, is composed of points in $\mathbb{R}^2$ where 250 of the x_i values are distributed according to a bivariate normal distribution $\mathcal{N}((0,0), \mathbf{I}0.125^2)$ and the remaining 250 points are distributed according to $\mathcal{N}((0.5,0), \mathbf{I}0.125^2)$. The points form two circular clusters whose centers are 4 standard deviations apart, Figures 2 (a) and (d) show a visualization. We use these two distributions to emphasize the differences between the REVI policy that accounts for the task correlation and the NEVI and EVI policies that do not. We use 500 points so that the underlying task distribution is truly represented and differences in opportunity cost of policies are not simply due to this single realization of the task distribution.

We perform experiments with $A = 3, 5, 8$ tools. For each experiment in each set of tasks, we generate 8 vectors of true performance values, $\zeta_1, \dots, \zeta_8 \in \mathbb{R}^{500}$, and use only the first 3 or first 5 when $A = 3, 5$. Each performance vector, ζ_a, was randomly generated by sampling from a discretized Gaussian Process with a squared exponential kernel, $\zeta \sim \mathcal{N}(\underline{0}, \Sigma)$ where $\Sigma_{ij} = \sigma_0 \exp(-D(x_i - x_j, l_1, l_2)/2)$ and $D(x_i - x_j, l_1, l_2) = (x_{i,1} - x_{j,1})^2/l_1^2 + (x_{i,2} - x_{j,2})^2/l_2^2$. The parameters for the Gaussian Process generating the data were $\sigma_0 = 1$, and $l_1 = l_2 = 0.1$, the same hyperparameters were used for all generated data and both task sets. The variance of the added noise was set to $\sigma^2_{\epsilon,a} = 1/10^2$ for all tools, and noise is independently and identically distributed for each sample. Figures 2 (b),(e) show surface plots of the one of the sets of generated latent functions when $A = 3$.

We initialize each sampling procedure with 20 measurements per tool, 10 per dimension as recommended by Jones et al. (1998), allocated to tasks by a Latin Hypercube Design described in Section 5.2. After the initialization, a Gaussian Process is fitted and $\mu_1^{20}, ..., \mu_A^{20}$ and $\Sigma_1^{20}, ..., \Sigma_A^{20}$ are calculated. We separately apply REVI, NEVI and EVI sequential policies until the budget of $N = 300, 500, 800$ has been consumed for experiments with 3, 5 and 8 tools respectively. For comparison we also construct mappings using samples allocated by Latin Hypercube Designs with equivalent budgets $N = 20A$ to $N = 100A$.

All reported results are averaged over 400 replications, with 400 different sets of performance vectors. For each set of vectors, every sampling policy was initialized with the same Latin Hypercube Design and the same random number stream for noise values. At each stage during sampling, the mapping is constructed by choosing the highest predicted tool for each task, $S(x_i) = \underset{a}{\text{argmax}}\ \mu_{i,a}^n$, and the true opportunity cost of the mapping is measured,

$$OC = \sum_i \max_a \zeta_{i,a} - \zeta_{i,S(x_i)}.$$

Figure 3 shows how the average opportunity cost reduces with the number of samples taken, averaged over the 400 runs. Table 1 gives the final average Opportunity Cost with standard errors.

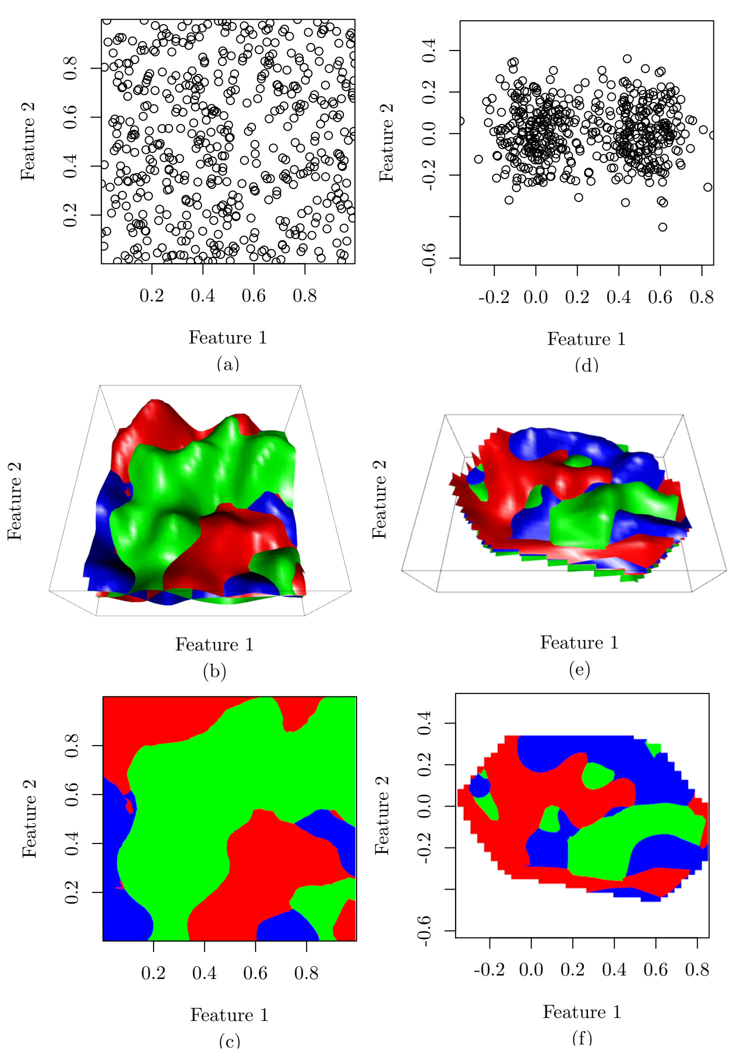

Figure 2: The left column is for uniform problem instances and the right is for bimodal. Figures (a),(d) show the sets of task features with randomly distributed values X_U (a) and X_B (d). (b),(e) show one example of the perforamnce surfaces (generated by a bicubic spline interpolation) for three tools ζ_1, ζ_2, ζ_3. (c),(f) show the true optimal mapping from features to best of the three tools $S(x) = \underset{a}{\operatorname{argmax}}\ \zeta_{a,i}$.

5.2. Mapping based on Latin Hypercube Design

Given a sampling budget that is a multiple of A, we allocate $N_{LHD} = N/A$ samples to each tool. N_{LHD} tasks are chosen from the set of 500 by a Latin Hypercube applied to the ranks of the sorted feature values $X \subset \mathbb{R}^2$. This makes no difference for X_U as the ranks and feature values are both uniformly distributed. However for X_B, an LHD applied to the feature values would undersample task dense regions and oversample sparse regions. Applying the LHD to the ranks results in hypercube divisions that are narrower/wider in dense/sparse regions. The tasks with a rank nearest to the Latin Hypercube points are selected to be included in the design. As with the sequential methods, a Gaussian Process with the squared exponential kernel is used to predict the expected performance at all tasks. The best predicted tool is chosen for each task in the mapping and the true opportunity cost is then measured. A new random design is chosen for every new budget and the performance predictions are re-calculated therefore this is not a sequential method.

5.3. Results

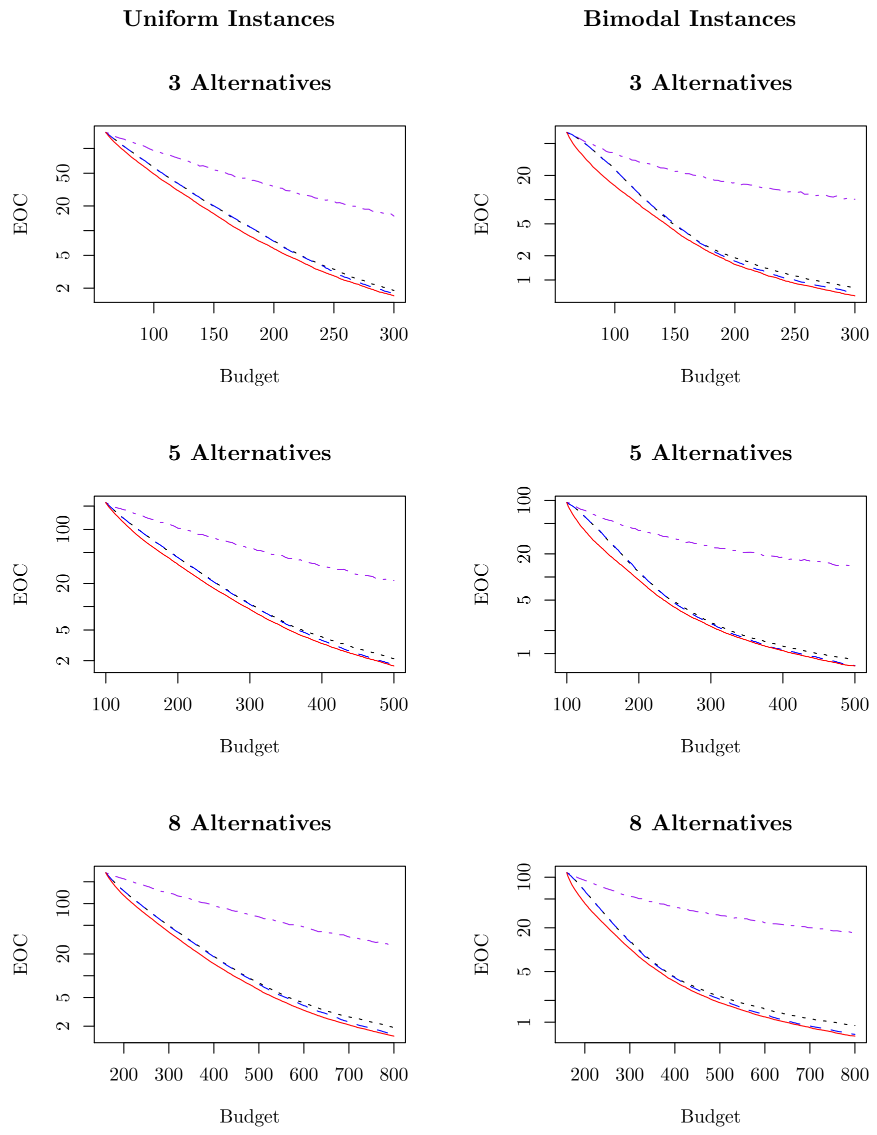

Figure 3: Opportunity cost of different sampling policies for various budget sizes averaged over 400 runs. In all plots: pink (dot-dash) is Latin Hypercube, black (dotted) is EVI, blue (dashed) is NEVI, and red (solid) is REVI. For all budgets, number of tools and feature distributions, the REVI policy produced the designs with the lowest opportunity cost on average.

Table 1: Final Average Opportunity Cost ± std. err. for different sampling policies. *Random* is the performance of a mapping that picks a random tool for each task. *Single Best* is the performance of the single truly best tool.

	Uniform		
Tools	3	5	8
Random	427.6 ± 0.06	583.8 ± 0.06	712.3 ± 0.07
Single Best	325.8 ± 0.31	448.4 ± 0.4	542.6 ± 0.46
LHD	15.06 ± 0.31	21.95 ± 0.4	26.44 ± 0.46
EVI	1.87 ± 0.06	2.12 ± 0.06	1.92 ± 0.07
NEVI	1.69 ± 0.05	1.78 ± 0.04	1.54 ± 0.04
REVI	1.61 ± 0.04	1.71 ± 0.04	1.46 ± 0.03

	Bimodal		
	3	5	8
Random	430.3 ± 0.03	583.4 ± 0.03	710.6 ± 0.04
Single Best	272.7 ± 0.23	377 ± 0.24	455 ± 0.25
LHD	10.13 ± 0.23	14.11 ± 0.24	17.1 ± 0.25
EVI	0.8 ± 0.03	0.83 ± 0.03	0.9 ± 0.04
NEVI	0.69 ± 0.02	0.7 ± 0.02	0.68 ± 0.02
REVI	0.63 ± 0.02	0.69 ± 0.02	0.64 ± 0.02

Figure 3 compares the opportunity cost for various budget sizes for different sampling policies and both task feature distributions. On average, the REVI policy provides the best mapping for both task distributions and all budget sizes. The NEVI and EVI policies make the assumption that maximizing the single task marginal expected improvement also maximizes the marginal expected mapping improvement. This may be approximately true in the uniform case where the effects of correlation are similar for most tasks. However this assumption is less true in the bimodal case where there is greater variation in task density and therefore the expected improvement due to covariance varies more between tasks. In the bimodal case, after the initial design, we see a divergence in average opportunity cost between the REVI policy and the NEVI or EVI policies. At the initial stages NEVI and EVI are more likely to allocate samples to unsampled outlying tasks providing smaller gains to portfolio performance, whereas samples allocated by REVI or the Latin Hypercube do account for task density. After the outliers have been sampled, the efficiency of NEVI and EVI improves.

EVI assumes that the noise variance is negligible compared to the posterior variance for the (task, tool) pair that maximizes the expected improvement. This assumption becomes less true as the budget size increases and posterior variance for even the maximizing (task, tool) pair reduces and noise becomes non-negligible. Therefore in all cases we see a slight divergence in the Average OC between EVI and NEVI for large budget sizes.

The final opportunity cost and standard errors are reported in Table 1. As the number of tools increases, the opportunity cost for LHD increases whereas the sequential policies do not increase suggesting that the policies given here scale with the number of tools and budget size much more favourably than the non-sequential design. In all cases the REVI policy produced the best performing mappings, and all policies were significantly better than the Latin Hypercube Designs with equivalent budget.

Given the final opportunity cost of the Latin Hypercube Design with budget sizes of 300, 500 and 800 when using 3, 5 and 8 tools respectively, the percentage budget reduction of the REVI policy to achieve the same level of opportunity cost was 49%, 53% and 58% in the X_U case and in the X_B case was 62%, 65% and 67% respectively.

6. Conclusion and Future Work

In this article, we extended the typical ranking and selection problem such that the performance of an alternative/tool may be described as a function over some input feature space, and the goal is to efficiently learn which tool performs best for each of a given set of tasks characterized by points in feature space. This has many applications, including algorithm selection, where we are given a set of problem instances and would like to learn which algorithm is best for each problem instance. Or in personalized medicine where one must efficiently identify a mapping from patient characteristics to most effective treatment.

We proposed the Regional Expected Value of Improvement (REVI) policy which samples in a way that maximizes the expected increase in predicted performance over all the tasks. This method is myopically optimal by construction and asymptotically optimal. We also proposed the NEVI and EVI sampling strategies that make some simplifying assumptions and no longer have the myopic optimality property, however they reduce the computatinoal complexity and memory requirement and significantly performed better than Latin Hypercube Design in our experiments.

Future work is to adapt the REVI policy to the case where samples do not consume the same amount of budget, for example if one has a limited time budget and the time taken to collect a sample varies depending on the task and tool. Another further possible development is the case when the set of tasks is not finite, but there is a continuous distribution of tasks.

References

Bingham, D., Ranjan, P., and Welch, W. J. (2014). Design of computer experiments for optimization, estimation of function contours, and related objectives. In Lawless, J. F., editor, *Statistics in Action: A Canadian Outlook*, chapter 7. CRC Press.

Branke, J., Chick, S. E., and Schmidt, C. (2007). Selecting a Selection Procedure. *Management Science*, 53(12):1916–1932.

Chen, C.-H., Chick, S. E., Lee, L. H., and Pujowidianto, N. A. (2015). Ranking and selection: Efficient simulation budget allocation. In *Handbook of Simulation Optimization*, pages 45–80. Springer.

Chick, S. E., Branke, J., and Schmidt, C. (2010). Sequential sampling to myopically maximize the expected value of information. *INFORMS Journal on Computing*, 22(1):71–80.

Frazier, P., Powell, W., and Dayanik, S. (2009). The knowledge-gradient policy for correlated normal beliefs. *INFORMS Journal on Computing*, 21(4):599–613.

Frazier, P. I., Powell, W. B., and Dayanik, S. (2008). A knowledge-gradient policy for sequential information collection. *SIAM Journal on Control and Optimization*, 47(5):2410–2439.

Gupta, S. S. and Miescke, K. J. (1996). Bayesian look ahead one-stage sampling allocations for selecting the best population. *Journal of Statistical Planning and Inference*, 54(2):229–244.

Hager, W. W. (1989). Updating the inverse of a matrix. *SIAM review*, 31(2):221–239.

Huang, D., Allen, T. T., Notz, W. I., and Zeng, N. (2006). Global optimization of stochastic black-box systems via sequential kriging meta-models. *Journal of Global Optimization*, 34(3):441–466.

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.

Picheny, V., Ginsbourger, D., Roustant, O., Haftka, R. T., and Kim, N.-H. (2010). Adaptive designs of experiments for accurate approximation of a target region. *Journal of Mechanical Design*, 132(7):071008.

Press, W. H., Teukolsky, S. A., Vetterling, S. T., and Flannery, B. P. (1996). *Numerical Recipes in C*, volume 2. Cambridge University Press, Cambridge.

Rasmussen, C. E. and Williams, C. K. I. (2004). *Gaussian Processes for Machine Learning*. MIT Press.

Rice, J. R. (1976). The algorithm selection problem. *Advances in Computers*, 15:65–117.

Sacks, J., Welch, W. J., Mitchell, T. J., and Wynn, H. P. (2012). Design and analysis of computer experiments. *Statistical Science*, 4(4):409–423.

Scott, W., Frazier, P., and Powell, W. (2011). The correlated knowledge gradient for simulation optimization of continuous parameters using gaussian process regression. *SIAM Journal on Optimization*, 21(3):996–1026.

Smith-Miles, K. A. (2008). Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Computing Surveys*, 41(1):1–25.

Xu, Y., Trippa, L., Müller, P., and Ji, Y. (2014). Subgroup-based adaptive (SUBA) designs for multi-arm biomarker trials. *Statistics in Biosciences*, 8(1):159–180.