Bayesian Optimization for Sensor Set Selection

R. Garnett, M. A. Osborne and S. J. Roberts
Department of Engineering Science
University of Oxford
Oxford, OX1 3PJ, UK
{rgarnett, mosb, sjrob}@robots.ox.ac.uk

ABSTRACT

We consider the problem of selecting an optimal set of sensors, as determined, for example, by the predictive accuracy of the resulting sensor network. Given an underlying metric between pairs of set elements, we introduce a natural metric between sets of sensors for this task. Using this metric, we can construct covariance functions over sets, and thereby perform Gaussian process inference over a function whose domain is a power set. If the function has additional inputs, our covariances can be readily extended to incorporate them—allowing us to consider, for example, functions over both sets and time. These functions can then be optimized using Gaussian process global optimization (GPGO). We use the root mean squared error (RMSE) of the predictions made using a set of sensors at a particular time as an example of such a function to be optimized; the optimal point specifies the best choice of sensor locations. We demonstrate the resulting method by dynamically selecting the best subset of a given set of weather sensors for the prediction of the air temperature across the United Kingdom.

Categories and Subject Descriptors

I.2.9 [**Computing Methodologies**]: Artificial IntelligenceRoboticsSensors

General Terms

Algorithms, Experimentation, Theory

Keywords

Bayesian methods, Gaussian processes, global optimization, experimental design, spatial learning, sensor networks, sensor selection, sampling design

1. INTRODUCTION

Consider the problem of selecting the optimal locations for making measurements of some dynamic spatial process. This problem, studied under the name of *spatial sampling design* [8, 14] is encountered in a wide range of applications, including soil science, petroleum geology, and oceanography. In particular, we consider the problem of placing a small number of sensors with the goal of making accurate global predictions about a spatial field with temporal variation, such as air temperature.

We do this by treating the prediction quality of a set of sensors as a "black-box"—a measure of the efficacy of a particular sensor layout that is supplied to us after having tried it. This measure, henceforth referred to as the *objective function*, could be any arbitrarily complicated function, and might potentially be provided to us by an external agency, such as a user of the sensor network. We then use these measurements of the objective function in order to perform inference about its likely value for other sensor sets. By balancing the resulting exploration–exploitation trade off, we can carefully select the sensor placements to be evaluated, mindful that we will usually only be able to afford a limited number of trials.

Other approaches [6] typically assume greater prior knowledge of how different sensor sets will perform, and are unable to fully exploit the information yielded by any experimental trials of sensor layouts. In contrast, our formulation allows us great flexibility, allowing the use of as much or as little knowledge that may exist, and permitting its application to problems of many different types. In its full generality, our method aims to optimize any function over point sets.

Our inference is enabled by the use of Gaussian processes, described in Section 2. In particular, we wish to use Gaussian processes to perform inference about how our objective function will vary over sets. We then describe Gaussian process global optimization in Section 3, which will allow us to optimize our objective function. In order to allow the proposed inference and optimization within a Gaussian process framework, however, we must specify a covariance function over sets. TO enable the construction of such a covariance, in Section 4 we define a metric over sets of sensors. Section 5 describes in more detail how we can perform the optimization of an objective function over sensor sets, and Section 6 contains the specifics of the experiments that validate our approach.

2. GAUSSIAN PROCESSES

Gaussian processes (GPs) offer a powerful method to perform Bayesian inference about functions [12]. A GP is defined as a distribution over the functions $X \to \mathbb{R}$ such that the distribution over the possible function values on any finite subset of X is multivariate Gaussian. For a function $y(x)$, the prior distribution over its values $\boldsymbol{y}$ on a subset

IPSN'10, April 12–16, 2010, Stockholm, Sweden.

$\boldsymbol{x} \subset X$ is completely specified by a mean vector $\boldsymbol{\mu}(\boldsymbol{x})$ and covariance matrix $\mathbf{K}(\boldsymbol{x}, \boldsymbol{x})$,

$$p(\,\boldsymbol{y} \mid I\,) \triangleq \mathrm{N}(\boldsymbol{y}; \boldsymbol{\mu}(\boldsymbol{x}), \mathbf{K}(\boldsymbol{x}, \boldsymbol{x})) \triangleq \frac{1}{\sqrt{\det 2\pi\, \mathbf{K}(\boldsymbol{x}, \boldsymbol{x})}} \exp\left(-\frac{1}{2}\left(\boldsymbol{y} - \boldsymbol{\mu}(\boldsymbol{x})\right)^{\mathsf{T}} \mathbf{K}(\boldsymbol{x}, \boldsymbol{x})^{-1} \left(\boldsymbol{y} - \boldsymbol{\mu}(\boldsymbol{x})\right)\right).$$

Here I, the *context*, includes prior knowledge of both the mean and covariance functions, which generate $\boldsymbol{\mu}$ and $\mathbf{K}$ respectively. We will incorporate knowledge of relevant functional inputs, such as x, into I for notational convenience. The prior mean function is chosen as appropriate for the problem at hand (often a constant), and the positive-definite covariance function is chosen to reflect any prior knowledge about the structure of the function of interest. For example, we can select covariance functions to reflect an expected degree of smoothness, periodicity, or isotropy. We can even specify covariance functions that explicitly acknowledge the possibility of changepoints or sensor faults [3].

We consider the family of covariance functions of the form

$$K(x_1, x_2; \lambda, \sigma) \triangleq \lambda^2 \kappa\big(d(x_1, x_2; \sigma)\big)\,, \tag{1}$$

where κ is an appropriately chosen function and d is a metric parametrized by σ. For one-dimensional inputs x, we will often use

$$d_{\text{simple}}(x_1, x_2; \sigma) \triangleq \frac{|x_1 - x_2|}{\sigma}\,. \tag{2}$$

In such a context, the parameters λ and σ represent respectively the characteristic *output* and *input scales* of the process. Example covariance functions of the form (1) are the squared exponential covariance, given by

$$K_{\text{SE}}(x_1, x_2; \lambda, \sigma) \triangleq \lambda^2 \exp\Big(-\frac{1}{2} d(x_1, x_2; \sigma)^2\Big)\,, \tag{3}$$

the rational quadratic covariance, parametrized by α,

$$K_{\text{RQ}}(x_1, x_2; \lambda, \sigma, \alpha) \triangleq \lambda^2 \left(1 + \frac{1}{2\alpha} d(x_1, x_2; \sigma)^2\right)^{-\alpha}, \tag{4}$$

and the the Matérn class of covariance functions, which are parametrized by a "roughness" ν, and for the example of $\nu = 3/2$ can be written

$$K_{\text{Mtn}}(x_1, x_2; \lambda, \sigma, \nu = 3/2) \triangleq \lambda^2 \left(1 + \sqrt{3}\, d(x_1, x_2; \sigma)\right) \exp\left(-\sqrt{3}\, d(x_1, x_2; \sigma)\right). \tag{5}$$

λ, σ, α and ν are examples of the set of *hyperparameters*, collectively denoted as θ, that are required to specify our covariance and mean functions. Other covariance functions can be constructed for a wide variety of problems [12]. For this reason, GPs are ideally suited for time-series–prediction problems with complex behavior.

Note that we typically do not receive observations of y directly, but rather of potentially corrupted versions z of y. In this paper, we only consider the Gaussian observation likelihood $p(\,\boldsymbol{z} \mid \boldsymbol{y}, \theta, I\,)$, although other observation likelihoods can be used [12]. In particular, we often assume independent Gaussian noise contributions of a fixed variance η^2. This noise variance effectively becomes another hyperparameter of our model and will therefore be incorporated into θ. To proceed, we define

$$V(t_1, t_2; \theta) \triangleq K(t_1, t_2; \theta) + \eta^2\, \delta(t_1 - t_2)\,, \tag{6}$$

where $\delta(\cdot)$ is the Kronecker delta function. Of course, in the noiseless case, $z = y$ and $V(t_1, t_2; \theta) = K(t_1, t_2; \theta)$. We define the set of observations available to us as $(\boldsymbol{x}_d, \boldsymbol{z}_d)$. Conditioning on these observations, I, and θ, we are able to analytically derive our predictive equations for the vector of function values $\boldsymbol{y}_\star$ at inputs $\boldsymbol{x}_\star$

$$p(\boldsymbol{y}_\star | \boldsymbol{z}_d,\, \theta,\, I) = \mathrm{N}\big(\boldsymbol{y}_\star; \boldsymbol{m}(\boldsymbol{y}_\star \,|\, \boldsymbol{z}_d,\, \theta,\, I)\,, \mathbf{C}(\boldsymbol{y}_\star \,|\, \boldsymbol{z}_d,\, \theta,\, I)\big)\,, \tag{7}$$

where we have

$$\begin{aligned} &\boldsymbol{m}(\boldsymbol{y}_\star \,|\, \boldsymbol{z}_d,\, \theta,\, I) \triangleq \\ &\boldsymbol{\mu}(\boldsymbol{x}_\star; \theta) + \mathbf{K}(\boldsymbol{x}_\star, \boldsymbol{x}_d; \theta)\mathbf{V}(\boldsymbol{x}_d, \boldsymbol{x}_d; \theta)^{-1}(\boldsymbol{z}_d - \boldsymbol{\mu}(\boldsymbol{x}_d; \theta)) \\ &\mathbf{C}(\boldsymbol{y}_\star \,|\, \boldsymbol{z}_d,\, \theta,\, I) \triangleq \\ &\mathbf{K}(\boldsymbol{x}_\star, \boldsymbol{x}_\star; \theta) - \mathbf{K}(\boldsymbol{x}_\star, \boldsymbol{x}_d; \theta)\mathbf{V}(\boldsymbol{x}_d, \boldsymbol{x}_d; \theta)^{-1}\mathbf{K}(\boldsymbol{x}_d, \boldsymbol{x}_\star; \theta)\,. \end{aligned} \tag{8}$$

Of course, we can rarely be certain about θ *a priori*. These hyperparameters must therefore be assigned an appropriate prior distribution and then marginalized. Although the required integrals are non-analytic, we can efficiently approximate them using Bayesian Monte Carlo [11] techniques. This entails evaluating our predictions for a range of hyperparameter samples $\{\theta_i : i \in S\}$. Each sample has an associated predictive mean $\boldsymbol{m}(\boldsymbol{y}_\star \,|\, \boldsymbol{z}_d,\, \theta_i,\, I)$ and covariance $\mathbf{C}(\boldsymbol{y}_\star \,|\, \boldsymbol{z}_d,\, \theta_i,\, I)$, which are then combined in a weighted mixture

$$\begin{aligned} p(\,\boldsymbol{y}_\star \mid \boldsymbol{z}_d,\, I\,) &= \frac{\int p(\,\boldsymbol{y}_\star \mid \boldsymbol{z}_d, \theta, I\,)\, p(\,\boldsymbol{z}_d \mid \theta, I\,)\, p(\,\theta \mid I\,)\, \mathrm{d}\theta}{\int p(\,\boldsymbol{z}_d \mid \theta, I\,)\, p(\,\theta \mid I\,)\, \mathrm{d}\theta} \\ &\simeq \sum_{i \in S} \rho_i\, \mathrm{N}\big(\boldsymbol{y}_\star; \boldsymbol{m}(\boldsymbol{y}_\star \,|\, \boldsymbol{z}_d,\, \theta_i,\, I)\,, \mathbf{C}(\boldsymbol{y}_\star \,|\, \boldsymbol{z}_d,\, \theta_i,\, I)\big)\,, \end{aligned} \tag{9}$$

with weights $\boldsymbol{\rho}$ as detailed in [10]. We also use the sequential formulation of a GP given by [10], a natural fit for our sequential decision problem. After each new function evaluation, we can efficiently update our predictions in light of the new information received.

3. GAUSSIAN PROCESSES FOR GLOBAL OPTIMIZATION

We now frame global optimization as a sequential decision problem [9]. Imagine that we have an unknown and expensive-to-evaluate objective function $y(x)$ that we seek to minimize. The cost associated with computing $y(x)$ compels us to select the location of each new evaluation very carefully. For example, in the case of a sensor network, the task of moving sensors to new locations might be very expensive in terms of time, effort, or monetary cost. Our problem's ultimate task is to return a final point x_M in the domain, and we define the loss associated with this choice to be equal to $y_M \triangleq y(x_M)$. At each iteration of the algorithm, we evaluate $y(x)$ at the point that minimizes our expected loss; that is, we aim to minimize our ultimate returned value y_M.

In particular, we wish to minimize a noisy objective function that changes over time. The problem we consider is one

in which we have a pre-defined time t_M at which we must return our minimum (x_M, y_M). Until that time, we have a certain number of exploratory evaluations to make. We may be able to select the times of our evaluations, or we may be forced to evaluate at constrained timesteps. Either way, we cannot choose an evaluation that is before any evaluation we have already taken—we cannot travel back in time. Time is now simply treated as an additional input to our objective function, meaning that our inputs will now be of the form $[x, t]$.

Our next step is to take a GP over the values of y. Inference about functions is at the heart of optimization, made explicit by techniques of optimization that employ response surfaces or surrogates [4]. Our goal is to build a statistical picture of the function's overall form given our observations of it.

To allow for the dynamic nature of our objective function, we build a covariance function over the combined input space $[x, t]$. We also acknowledge any observation noise within our GP framework. In particular, we assume that we possess Gaussian-noise-corrupted evaluations and proceed by making the appropriate modifications to our covariance, as in (6).

The noise and dynamics in our objective function also impact upon our choice of the final point to return as the minimum. Rather than returning a noise-corrupted observed value z_M at the final chosen x_M, we allow our algorithm to return its best guess for the real value y_M at that point. We constrain the returned x_M to the set of actual evaluated points. At such points, we are likely to have only a small amount of uncertainty about the objective function. We further limit the candidate points to return to those where we are still reasonably confident about the objective: as we are optimizing a dynamic function, its current value at input x is likely to be different from an evaluation at x made in the past. We enforce this final constraint by stipulating that the returned value must be at an x at which we have made an evaluation not more than ϵ units of time in the past.

Imagine that we have only one allowed function evaluation remaining before we must report our inferred function minimum, at the final time t_M. If

$$([\boldsymbol{x}_d, \boldsymbol{t}_d], \boldsymbol{z}_d) \triangleq \big\{([x_j, t_j], z_j) : j \in d\big\}$$

are the evaluations of the function we have gathered so far, we define

$$\eta(\theta) \triangleq \min_{j \in d:\, t_j > t_M - \epsilon} m(y_j \,|\, \theta, \boldsymbol{z}_d, I) \ .$$

Given this, we can define the loss $\lambda(y)$ of evaluating the function this last time at x and its returning y as

$$\lambda(y;\, \theta) \triangleq \begin{cases} y; & y < \eta(\theta) \\ \eta(\theta); & y \geq \eta(\theta) \end{cases} . \tag{10}$$

That is, after having observed y, our loss is simply the new minimum of the set of observed points, $\min\big(y, \eta(\theta)\big)$, which we would report as y_M.

Given our GP over y, we can determine an analytic expression for the expected loss of selecting x given that we have observed $\boldsymbol{y}_d$ and have only one evaluation remaining:

$$\begin{aligned} &\Lambda(x \mid \boldsymbol{z}_d,\, I) \\ &\triangleq \frac{\iint \lambda(y;\theta)\, p(y|\boldsymbol{z}_d, \theta, I)\ p(\boldsymbol{z}_d|\theta, I)\ p(\theta|I)\ \mathrm{d}\theta\, \mathrm{d}y}{\int p(\boldsymbol{z}_d|\theta, I)\ p(\theta|I)\ \mathrm{d}\theta} \\ &= \sum_i \rho_i \mathrm{V}_i\,(x \mid \boldsymbol{z}_d,\, I) \ , \end{aligned} \tag{11}$$

where we we have marginalized over hyperparameters using (9), and defined

$$\begin{aligned} &\mathrm{V}_i\,(x \mid \boldsymbol{z}_d,\, I) \\ &\triangleq \eta_i \int_{\eta_i}^{\infty} \mathrm{N}(y; m_i, C_i)\ \mathrm{d}y + \int_{-\infty}^{\eta_i} y\, \mathrm{N}(y; m_i, C_i)\ \mathrm{d}y \\ &= \eta_i + (m_i - \eta_i)\, \Phi(\eta_i; m_i, C_i) - C_i\, \mathrm{N}(\eta_i; m_i, C_i) \ . \end{aligned}$$

Note that we have denoted the usual Gaussian cumulative distribution function as Φ, and abbreviated $\eta(\theta_i)$ as η_i, $m(y\,|\,\theta_i, \boldsymbol{z}_d, I)$ as m_i, and $C(y\,|\,\theta_i, \boldsymbol{z}_d, I)$ as C_i. The location where (11) is lowest gives the optimal location for our next function evaluation, balancing both exploration and exploitation. Any uncertainty about the model is dealt with through our principled marginalization of hyperparameters, influencing our selection of evaluations in a principled manner. Note that the "expected improvement" function of [5] is close to but differs from this Bayesian expected loss criterion.

Of course, we have merely shifted the minimization problem from one over the objective function $y(x)$ to one over the expected loss function (11). Fortunately, the expected loss function is computationally inexpensive to evaluate, much more so than the objective function.

The action of our algorithm is illustrated in Figure 1.

4. METRICS OVER SETS

We now consider the challenge of constructing covariance functions of the form (1) to perform inference about the unknown values of an objective function defined over point sets. We assume that the objective function is smooth, in the sense that it will have similar values for "related" sets. For example, if we are considering the predictive performance of a sensor network, we should expect *a priori* that moving each sensor by a small amount will typically not greatly affect its performance.

To formalize the meaning of "related" in this context, we define a distance between two sets of points in terms of an underlying distance between singletons. This distance can then be incorporated into an off-the-shelf covariance (1) for Gaussian processes in the optimization procedure. Suppose we have a domain X of interest with an associated parametrized metric $d'(x_1, x_2; \theta)$ defined on $X \times X$. We assume that the chosen metric would be useful for performing inference about the objective function if we restricted the domain to singleton sets, and extend this distance to arbitrary subsets of X. The distance d' can be readily obtained for sets of spatial coordinates; for example, the usual Euclidean distance (or great-circle distance on the earth) usually suffices.

The optimization we suggest for sensor selection in this paper does not require any particular form for the underlying prediction algorithm. If, however, a Gaussian process over the field measured by a sensor network (separate from the Gaussian process over sets used for optimization) is used

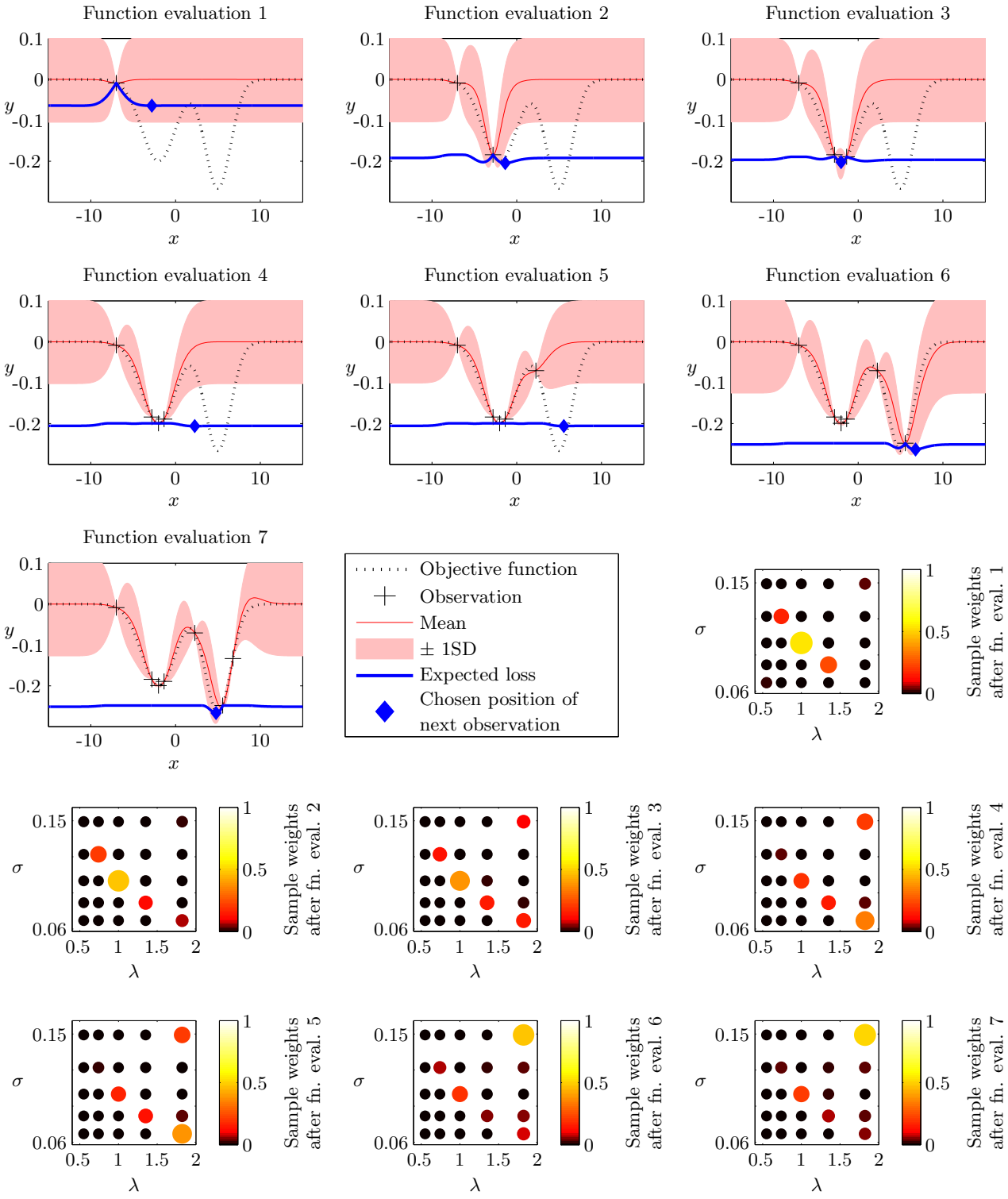


Figure 1: An illustration of our algorithm balancing the exploitation of the current local minimum and the exploration of unvisited areas. The chosen covariance was a squared exponential, giving us two hyperparameters: an input scale σ, and an output scale λ. As we proceed, evidence from our observations compels the sample weight to be transferred from the prior mean, at $\sigma = 1$ and $\lambda = 0.1$, towards larger and more representative scales. This gives rise to a smoother GP fit with larger error bars, which then influences the selection of future evaluations.

in this capacity, we have several more options for the distance d'. In such a case, the underlying prediction Gaussian process will have a parametrized covariance $K'(x_1, x_2; \sigma)$ defined on $X \times X$. If known, such a covariance encapsulates much information about the unknown field, information that could be employed in our determination of the distance between sets. This might simply involve the sharing of a characteristic length scale between K' and d'. If the covariance is more sophisticated, reflecting, for example, periodicity or changepoints in our field, we might instead build a pairwise distance as

$$d'(x_1, x_2; \theta) \triangleq \sqrt{K'(x_1, x_1; \theta) + K'(x_2, x_2; \theta) - 2K'(x_1, x_2; \theta)}\,.$$

The covariance K' might also be used to directly define a distance over sets in a manner that is independent of any intermediate definition of a pairwise distance d'.

Returning to the general problem, we seek to define a parametrized metric $d(A, B; \theta)$ for $A, B \subseteq X$. We write $A \triangleq \{a_i;\, i = 1, \ldots, n\}$ and $B \triangleq \{b_j;\, j = 1, \ldots, m\}$.

This metric between sets can be selected as appropriate for the task at hand. Below, we propose a particular metric between sets that can be used when the objective function considered is related to the predictive accuracy of a sensor network. Before we introduce the metric, we first introduce some examples, depicted in Figure 2, to motivate our choice.

Figure 2a illustrates the first intuition to which we appeal—if $d'(a_i, b_j; \theta)$ is large for all $a_i \in A$ and $b_j \in B$, we should not expect the evaluation of the objective function at A to be strongly correlated with its evaluation at B. In such a case, $d(A, B; \theta)$ should be large.

Next, Figure 2b illustrates the case in which $d'(a_i, b_j; \theta)$ is large for a single $b_j \in B$ and every $a_i \in A$. In this case, we also want $d(A, B; \theta)$ to be somewhat large. That single b_j may be in a location such that $f(B)$ will be significantly different from $f(A)$.

Figure 2c has, for every $a_i \in A$, $d'(a_i, b_j; \theta)$ small for some $b_j \in B$. In such a case, $d(A, B; \theta)$ should be small, because A and B, despite relabeling, have very similar locations overall (according to d'). For determining the distances between A and B in this case, the distance should depend solely on the proximity of each point in A to the *closest* point in B. It is additionally clear that any permutation of sensor labels should never be important.

It seems reasonable that adding a point very close to an existing point should not dramatically change the performance of a set. Figure 2d illustrates the sets from Figure 2c with a_3 added close to a_2 and b_3 added close to b_2. In such a case, the distance between A and B should remain close to that from Figure 2c.

Finally we attempt to draw out our intuitions for how the distance should behave over sets of unequal size. Figure 2e has $d'(a_1, b_j; \theta)$ large for all $b_j \in B$, and $d'(b_i, b_j; \theta)$ small for all b_i, $b_j \in B$. In this case,

$$d(\{a_1\}, \{b_1\}; \theta) \simeq d(\{a_1\}, \{b_1, b_2\}; \theta)\,.$$

Given that b_1 and b_2 are very close, the addition of b_2 should not dramatically change the distance between A and B.

We now propose a metric that satisfies these desiderata.

The metric to which we appeal is the earth mover's distance (EMD), which is well-known and widely used in image processing [13, 7]. The earth mover's distance is defined for two "signatures" of the form $\{(x_i, w_i)\}$, where the $\{x_i\}$ are points in space (in our case, points on the sphere S^2), and the $w_i \in \mathbb{R}^+$ are positive real weights. When the total weight of each signature sums to unity (that is, each signature represents a discrete probability distribution), the EMD is equivalent to the first Wasserstein or Mallows distance [7]. We will assume henceforth that each signature normalizes in this manner.

Given two signatures $A \triangleq \{(a_i, w_i)\}_{i=1}^{n}$ and $B \triangleq \{(b_j, v_j)\}_{j=1}^{m}$, the EMD is defined intuitively as the minimum amount of work required to move the points in A to be aligned with the points in B. Here the amount of work for moving a point is proportional to its weight and the distance traveled. Considering each signature as a collection of mounds of dirt, the EMD is the minimum amount of earth (times distance) one would have to move to transform one signature into the other, hence its name.

More formally, the EMD is the optimum of the following linear program [13]:

$$d_{\mathrm{EMD}}(A, B) \triangleq \min \sum_i \sum_j f_{ij}\, d'(a_i, b_j) \tag{12}$$

subject to the following constraints

$$\begin{aligned} f_{ij} &\geq 0 && (1 \leq i \leq n, 1 \leq j \leq m) \\ \sum_j f_{ij} &= w_i && (1 \leq i \leq n) \\ \sum_i f_{ij} &= v_j && (1 \leq j \leq m) \\ \sum_i \sum_j f_{ij} &= 1, \end{aligned}$$

where $d'(a_i, b_j)$ is the underlying pairwise distance discussed previously. The solution to this optimization problem can be found in polynomial time ($O(n^3)$ when $|A| = |B| = n$) using the well-known Hungarian algorithm for transportation problems.

In the case of sensor networks, corresponding two sets of sensors to the "signatures" discussed above is mostly straightforward, except the assignment of weights to each point. Naïvely we might try a simple uniform weighting for each point in each set. The resulting distance achieves nearly all the heuristic goals set forth in the previous section. In particular, the uniformly weighted earth mover's distance has the desired behavior for Figures 2a, 2b, 2c, and 2e. Figure 2d presents a problem, however: using a uniform weighting, the optimal flow pairs the two closest opposite-colored points at the top, b_1 and a_2, and similarly the two closest opposite-colored points at the bottom, a_1 and b_2. The remaining two points, a_3 and b_3, are then compelled to share an edge of length δ, and the overall distance will be approximately $\delta/3$, assuming the vertical distance dominates the horizontal. This behavior is not to be desired; simply by increasing the vertical distance, we can arbitrary increase the distance between these two sets, but their performance should be expected to be similar for any distance δ.

In this case, of course, not all sensors are created equal. Assuming sufficient range, and isotropy, the sensors a_2 and a_3 will have significant overlap, as will b_2 and b_3. In the limit where a_2 and a_3 lie completely on top of each other, they might as well be a single sensor, as one has been made completely redundant. With this insight, we modify the

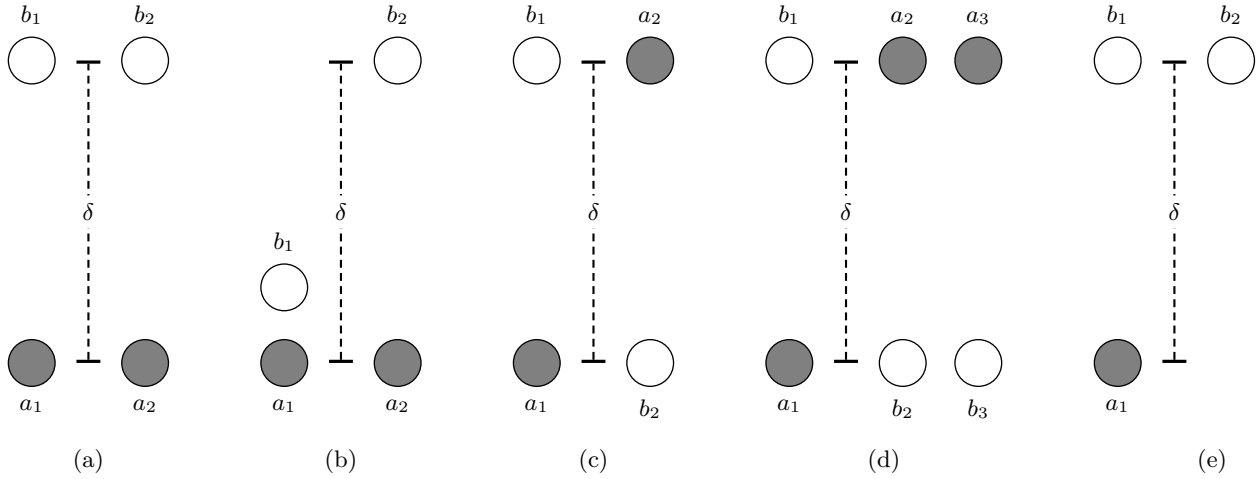


Figure 2: For δ sufficiently large, the distance $d(A, B; \theta)$ between sets A and B is (a) large, (b) somewhat large, (c) small, (d) small and (e) large.

weights for the two signatures, such that the nearby sensors a_2 and a_3 share nearly half the total weight of the A set (as they, combined, act essentially as a single sensor), and b_2 and b_3 half the total weight of the B set. With this modification, the EMD will now pair both a_2 and a_3 with b_1, both sets now each having nearly the same weight. The overall distance will now be small (and in the limit not depend on δ), as expected.

Depending on the nature of the sensors and the prediction scheme used, any number of weighting schemes taking this notion of redundancy into account could be used; see, for example, [2] for a discussion of redundancy in sensor networks with identical isotropic sensors. The chosen scheme might also incorporate further information about the range of each sensor; for example, if sensors a_2 and a_3 were actually directional sensors pointing away from each other, it would be inappropriate to reduce their weighting as discussed above as they now have very little overlap, despite their proximity.

We offer an appropriate weighting scheme when the underlying prediction algorithm is implemented using a Gaussian process, with covariance K'. We imagine using observations $\boldsymbol{z}_A$ of that field taken at at locations given by our set elements, $\boldsymbol{x}_A$, in order to make predictions about the field elsewhere, $\boldsymbol{x}_\star$. The mean prediction is then given by that in (8), a weighted sum of those observations,

$$\boldsymbol{\mu}(\boldsymbol{x}_\star) + \mathbf{K}'(\boldsymbol{x}_\star, \boldsymbol{x}_A)\mathbf{V}'(\boldsymbol{x}_A, \boldsymbol{x}_A)^{-1}(\boldsymbol{z}_A - \boldsymbol{\mu}(\boldsymbol{x}_A))\,.$$

With this in mind, we take the weights associated with those observations as weights on their locations, the set elements themselves. Rather than choosing a single set of observations, we will average over all possible $\boldsymbol{x}_\star$, covering every point in the domain. By an appeal to symmetry, integrating over the entire (infinite) domain forces the average value $\mathbf{K}'(\boldsymbol{x}_\star, \boldsymbol{x}_a)$ to be identical for every element $a \in A$. Therefore we take the weights on the elements of A to be

$$\boldsymbol{w}_A = \frac{\mathbf{1}_A^\mathsf{T}\,\mathbf{K}'(\boldsymbol{x}_A, \boldsymbol{x}_A)^{-1}}{\mathbf{1}_A^\mathsf{T}\,\mathbf{K}'(\boldsymbol{x}_A, \boldsymbol{x}_A)^{-1}\,\mathbf{1}_A}\,, \tag{13}$$

where $\mathbf{1}_A$ is a column vector of appropriate size whose every element is equal to one. Note that we have normalized our weights to unity.

These weights, the average weights obtained when making predictions given $\boldsymbol{x}_A$, naturally reflect the behaviour we require. Elements far away from all others will receive a large quantity of weight, whereas tight clusters of elements will essentially have the weight of a single element at that position shared amongst them. We use exclusively the EMD distance that results from the use of this weighting.

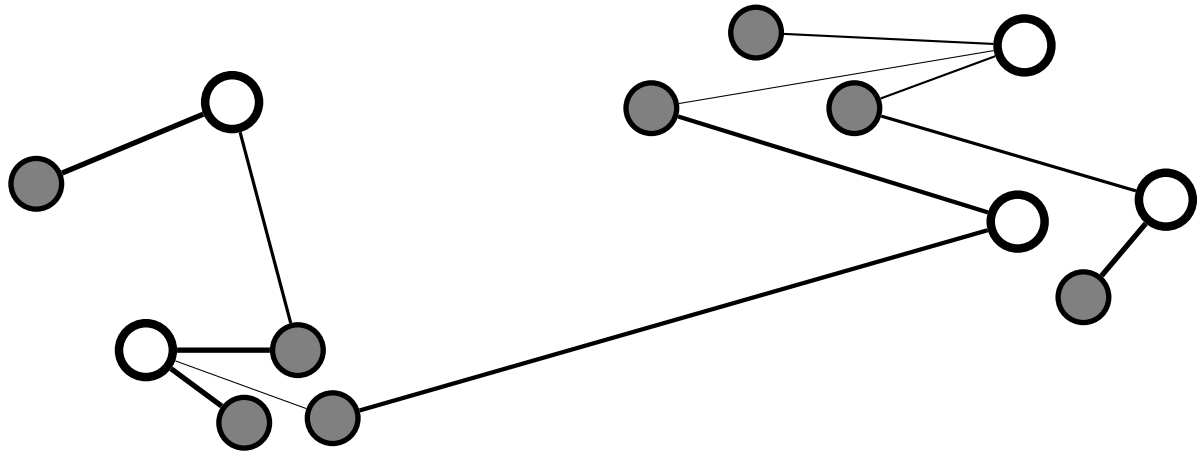
Figure 3 demonstrates the difference between the unweighted and weighted EMD distances. In the weighted version, clusters of nearby sensors receive appropriate weights, and the overall distance for the weighted version is more natural than the unweighted version, which is forced to resort to incorporate long, highly weighted, awkward edges.

5. ALGORITHM

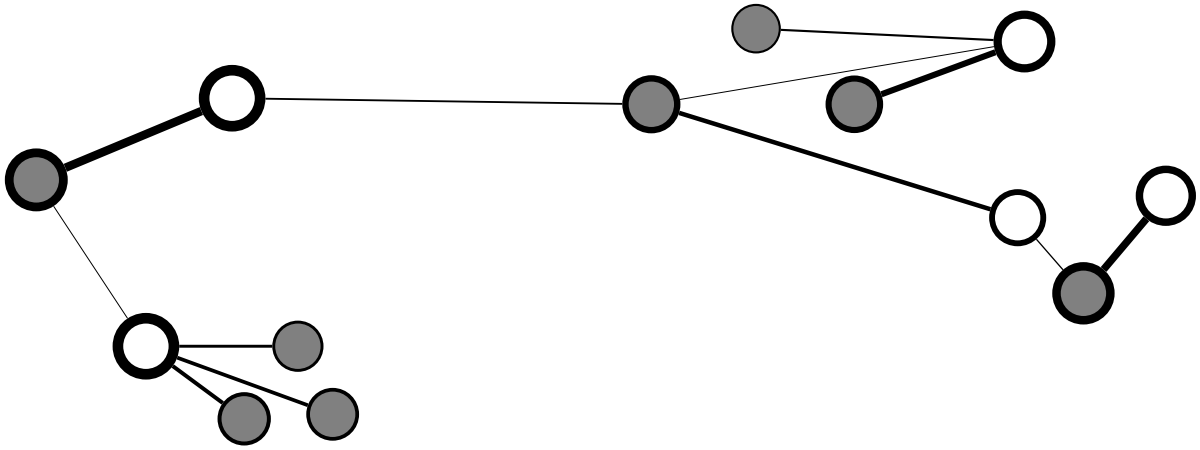
We now briefly describe how GPGO could be used for optimizing functions defined on a power set. Suppose we have a discrete set X and a real-valued function $f\colon \mathcal{P}(X) \to \mathbb{R}$ defined on the power set of X. Suppose further that we have selected an appropriate distance $d\colon \mathcal{P}(X) \times \mathcal{P}(X) \to \mathbb{R}^+$ between subsets of X. We use d to build a covariance of the form (1). Given this covariance, and an appropriate mean function over $\mathcal{P}(X)$, the application of GPGO to our problem is straightforward. At each step of the optimization process, we evaluate f on a particular subset of X. We update our GP over $\mathcal{P}(X)$ with this observation. Next we evaluate (11) on candidate subsets in $\mathcal{P}(X)$; the minimum among these becomes the subset used for the next evaluation of f. We continue in this manner for as long as desired.

In many cases, the objective function f might exhibit dynamism; that is, it may be a function of both time and the chosen subset. In this case, we adjust our inputs, as described in Section 3, to be of the form $[S(i), t(i)]$, where $t(i)$ is the time of the ith observation. As such, our method is able to cope with the potentially dynamic nature of our objective function, changing the selection of subset over time as appropriate. Pseudo-code for N iterations of our algorithm is provided in Algorithm 1.

Of course, the size of $\mathcal{P}(X)$ grows exponentially in the size of X, and evaluating (11) at every point quickly becomes



(a)



(b)

Figure 3: An example of the EMD between two sets, differentiated by their shading. The distance is the sum of the lengths of the depicted node connections, multiplied by the edge weighting, indicated here by the line-width of the edge. These edge weights are determined with reference to the node weights, similarly indicated by the linewidth of the node. For (a), the total (unit) weight for each set is evenly split between all nodes. For (b), the weight for each node is assigned according to its importance, using (13). Note that nodes that might be expected to be of greater importance to a sensor network, those that are far separated from other sensors in the network, are assigned greater weight.

infeasible. However, the nature of this expected loss function suggests various search heuristics that can be adopted to guide the search. In particular, the expected loss function is likely to be minimized at either a point that is close to an observation we have already made (exploitation), or far away from every observation (exploration). We can speed up our search by limiting our search to sets with these properties. Of course, evaluating the distance itself between all pairs of subsets can also become quickly infeasible, but we may apply further heuristics, for example encouraging exploitation by including subsets that differ from our current observations by one point and encouraging exploration by including a random selection of the remaining subsets. The effect to Algorithm 1 is that the search over the set S' is constrained to these heuristically chosen points.

Algorithm 1 Our method for selecting sensor subsets.

Require: Initial set $S(1)$.
  for $i = 1 \dots N$ **do**
    $f(i) \leftarrow f(x(i))$
    $S(i+1) \leftarrow$
      $\operatorname{argmin}_{S'} \Lambda\big([S', t(i+1)] \mid f(1), \dots, f(i), I\big)$
  end for
  Return: $S(N+1)$

5.1 Running time

Suppose that we have chosen σ hyperparameter samples. Incorporating the nth objective function evaluation into our optimization GP requires making rank-one updates to σ Cholesky factorizations of size $(n-1)^2$, and each training step thus runs in $O(hn^2)$ time. Evaluating the expected loss at k points takes time $O(khn^2)$. In practice, a somewhat inefficient implementation of the algorithm described above took approximately 120 seconds to train the GPGO model and select the next subset when trained on 100 observations, using approximately 1 000 hyperparameter samples and limiting the evaluation of the expected loss to 40 000 subsets. The machine was an Apple Mac Pro workstation with two 2.26 GHz quad-core Intel Xeon "Gainestown" processors (model E5520) and 16 GB of RAM, running the Linux kernel (version 2.6.31).

6. EXPERIMENTS

We tested our algorithm on the UK Meteorological Office MIDAS land surface stations data [1]. In particular, we chose a dispersed set of 50 sensors (whose locations are plotted in Figure 4) which recorded meteorological measurements for every day between 1959 and 2009, inclusive. The field we chose to perform prediction over is the maximum daily temperature.

Equipped with an appropriate distance between point sets for the task at hand, we describe our algorithm for optimizing the objective function over subsets of sensor locations. In this case, the domain X whose power set we wish to consider is the chosen set of 50 sensors. The function f to be optimized is a measure of the effectiveness of a selection of sensors. The particular measure of performance used in our experiments was the root mean squared error (RMSE) of the predictions made by using observations from the chosen sensors to predict the maximum temperature at every sensor location in the UK.

Specifically, by varying the sensor set, we aim to minimize the objective function $\mathrm{RMSE}(x; \Delta)$, where the inputs $x = [S, t_0]$ are an initial time t_0 and a set of sensors S. RMSE returns the root mean squared error of the predictions made about a relevant field at all available locations at the (discretized) set of test times from t_0 to $t_0 + \Delta$.

For the MIDAS data, we began a new trial at the beginning of each period of $\Delta = 28$ days. We select a set of 5 sensors, and then perform zero-step lookahead prediction for the temperatures at all 50 sensors at each of the subsequent 28 days. As a means of testing the effectiveness of our algorithm, we force our algorithm to choose its best subset every five years, and evaluate the performance of predictions made over the entire subsequent year.

Our predictions were made using a GP [10] sequentially trained on the readings at all test times. By this, we mean that the GP possesses observations from S at all available times in the range t_0 to $t_\star$, inclusive, when making predictions about that test time $t_\star$. That is, observations from the current time $t_\star$ are available in making predictions about other locations at that time. Prediction about future times (that is, the use of a non-zero lookahead) is, of course, possible if an application calls for such an objective function. If necessary, the prediction algorithm could also be granted a "burn-in" period so that the RMSE is not dominated by poor predictions made when very little data is available.

For the GP used to perform temperature predictions, we used a simple covariance that was a product of a term over time and a term over space. Both covariances were taken to be rational quadratics (4), the former using the simple distance d_{simple} from (2), and the latter using the great-circle distance between points on the surface of the earth. The hyperparameters of this GP were sequentially marginalized using the BMC procedure outlined in (9). The emphasis of this paper was not to make the most accurate predictions possible (for which we would recommend more sophisticated non-stationary covariances), but rather to perform sensor selection given any arbitrary black-box objective function.

The covariance function used for GPGO was taken as a product of a term over time and a term over sets. Both were taken as squared exponentials (3), the the former using the simple distance d_{simple} from (2). For the latter, we used our weighted EMD, d_{EMD} from (12) and (13). The hyperparameters of these covariances were sequentially marginalized using BMC, using (9). We applied search heuristics as suggested in Section 5, minimizing the expected loss at approximately 40 000 points for each selection. This includes all subsets that differ in only a single point from the current subset, along with many other, randomly chosen, subsets included for exploration.

We concentrate here on the selection of sets of a preselected, fixed size (5 sensors). We search over sets of that size when selecting the set at which our next evaluation of the objective function will be made.

To provide a comparison for our algorithm, we first tested against a simple random-selection algorithm. For this, subsets were randomly chosen for each trial period of 28 days. For each test year, we selected the subset that yielded the best RMSE performance in the preceding five years.

We also implemented the method outlined in [6], henceforth referred to as the mutual information (MI) algorithm. Using it, sensors were greedily selected (up to the allowed number of 5 sensors) to maximize the mutual information

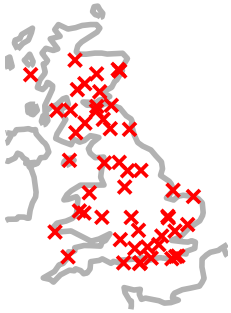


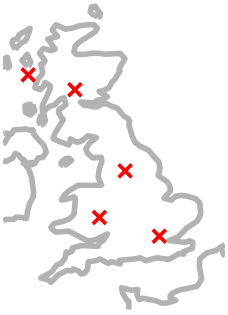
Figure 4: For our experiments over the MIDAS data, the locations of all 50 sensors.

(a) MI, 1964

(b) MI, 1979

(c) MI, 1994

(d) MI, 2009



(e) GPGO, 1964

(f) GPGO, 1979

(g) GPGO, 1994

(h) GPGO, 2009

Figure 5: For our experiments over the MIDAS data, the locations of sensors selected at the years indicated by (a)–(d), MI and (e)–(h), GPGO.

Table 1: The root mean squared error (in degrees Celsius) of the GP prediction algorithm using subsets selected by our subset optimization routine and the MI algorithm for tracking the weather in the UK for selected years.

	Method		
Year	Random	MI	GPGO
1964	2.9416	3.0182	**2.4408**
1969	3.0295	3.1553	**2.7702**
1974	2.6945	2.8208	**2.4310**
1979	2.8359	2.8238	**2.8143**
1984	2.9156	3.0690	**2.6249**
1989	2.9538	3.0845	**2.6438**
1994	3.0370	2.9886	**2.8252**
1999	3.0223	3.5081	**2.8561**
2004	2.9548	3.0695	**2.6886**
2009	3.0111	**2.8051**	2.8981
mean	2.9396	2.8256	**2.6993**

between the subsequently evaluated locations in space-time (that is, observations at the positions of the selected sensors, once a day for 28 days) and the locations of all of the other 50 sensors for each of the forthcoming 28 days. This mutual information was determined using the same GP used to provide predictions. Specifically, the MI method's selection was made according to the posterior covariance of this GP (in particular, its highest weighted hyperparameter sample).

The results of our algorithm are shown in Table 6. Overall, our algorithm was able to provide better subsets for the purposes of maximizing sensor network performance over time, providing sets with the best predictive accuracy for all bar one of the ten selected test years. As can be seen, our approach also yielded an overall RMSE that was significantly better than either tested alternative.

The subsets selected by both MI and our algorithm for a selected number of test years are displayed in Figure 5. All selected sets seem intuitively to provide good overall coverage of the UK. In that there is any deviation from this general spread of sensors, note that our algorithm selects a greater number on the west coast of the UK. It is possible that sensors in this region were determined to be useful for predicting temperatures via their good performance on the prediction task (weather predominantly moves from west-to-east in the UK). This nuance was automatically captured by our algorithm.

Our algorithm was able to explore the subset space reasonably well and locate reasonable optima, despite few observations (about 13 per year for 50 years from a total number of $\binom{50}{5} = 2\,118\,760$ possibilities). Additionally, our algorithm was able to adapt to information obtained by observing the performance of various subsets through time. Finally, by using a covariance defined on both space and time, our algorithm can cope with changes to the performance of various subsets over time (for example, if a sensor becomes faulty), allowing for us to react to *concept drift* in the prediction problem.

7. CONCLUSIONS

We introduce a novel algorithm for optimizing objective functions defined on point sets, and apply this algorithm to maximizing the predictive performance of sensor networks in dynamic environments. Starting with a chosen metric between single points, we define a useful metric between sets of points to be used in a Gaussian process covariance function. Using this distance over sets and the technique of Gaussian process global optimization, we optimize predictive performance as a function of chosen sensor set. By using a covariance function that is defined over time as well as subset space, our algorithm can dynamically adapt to changes in the prediction problem. An experiment using actual temperature data in the United Kingdom over 50 years demonstrated the efficacy of our algorithm for the chosen task.

Multiple extensions present themselves. Firstly, our algorithm could be used to manage a set of mobile sensors. Here it would be trivial to include a cost proportional to the total distance moved by all sensors into the GPGO loss function (11). We could also define a cost associated with taking an additional sensor, and allow our algorithm to search over sets of all sizes. In effect, our algorithm would select the optimal set size, trading off the cost of larger sets against the potentially better objective function values they may return. Such extensions to the cost function can easily be incorporated into our scheme due to our straightforward Bayesian approach.

8. ACKNOWLEDGMENTS

The work reported in this paper was funded by the Systems Engineering for Autonomous Systems (SEAS) Defence Technology Centre established by the UK Ministry of Defence. We thank the UK Meteorological office for the use of the MIDAS dataset.

9. REFERENCES

[1] British Atmospheric Data Centre. UK Meteorological Office MIDAS Land Surface Stations data (1853-current). Available from: `http://badc.nerc.ac.uk/data/ukmo-midas`, 2009.

[2] Y. Gao, K. Wu, and F. Li. Analysis on the redundancy of wireless sensor networks. In *Proceedings of the 2nd ACM international conference on wireless sensor networks and applications*, pages 108–114. ACM New York, NY, USA, 2003.

[3] R. Garnett, M. A. Osborne, S. Reece, A. Rogers, and S. J. Roberts. Sequential Bayesian prediction in the presence of changepoints and faults. *The Computer Journal*, 2010.

[4] D. R. Jones. A Taxonomy of Global Optimization Methods Based on Response Surfaces. *Journal of Global Optimization*, 21(4):345–383, 2001.

[5] D. R. Jones, M. Schonlau, and W. J. Welch. Efficient Global Optimization of Expensive Black-Box Functions. *Journal of Global Optimization*, 13(4):455–492, 1998.

[6] A. Krause, A. Singh, and C. Guestrin. Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. *The Journal of Machine Learning Research*, 9:235–284, 2008.

[7] E. Levina and P. Bickel. The earth mover's distance is the Mallows distance: Some insights from statistics. In *Proc. ICCV*, volume 2, pages 251–256. Citeseer, 2001.

[8] R. Olea. Sampling design optimization for spatial functions. *Mathematical Geology*, 16(4):369–392, 1984.

[9] M. A. Osborne, R. Garnett, and S. J. Roberts. Gaussian Processes for Global Optimisation. In *3rd International Conference on Learning and Intelligent Optimization (LION3)*, 2009. Available at `http://lion.disi.unitn.it/intelligent-optimization/LION3/online_proceedings/94.pdf`.

[10] M. A. Osborne, A. Rogers, S. Ramchurn, S. J. Roberts, and N. R. Jennings. Towards real-time information processing of sensor network data using computationally efficient multi-output Gaussian processes. In *International Conference on Information Processing in Sensor Networks (IPSN 2008)*, pages 109–120, April 2008.

[11] C. E. Rasmussen and Z. Ghahramani. Bayesian Monte Carlo. In S. Becker and K. Obermayer, editors, *Advances in Neural Information Processing Systems*, volume 15. MIT Press, Cambridge, MA, 2003.

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

[13] Y. Rubner, C. Tomasi, and L. Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2): 99–121, 2000.

[14] Z. Zhu and M. Stein. Spatial sampling design for prediction with estimated parameters. *Journal of Agricultural, Biological, and Environmental Statistics*, 11(1):24–44, 2006.