Real-Time Information Processing of Environmental Sensor Network Data using Bayesian Gaussian Processes[1]

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

and

A. Rogers and N. R. Jennings
School of Electronics and Computer Science
University of Southampton
Southampton, SO17 1BJ, UK.
{acr,nrj}@ecs.soton.ac.uk

In this paper, we consider the problem faced by a sensor network operator who must infer, in real-time, the value of some environmental parameter that is being monitored at discrete points in space and time by a sensor network. We describe a powerful and generic approach built upon an efficient multi-output Gaussian process (GP) that facilitates this information acquisition and processing. Our algorithm allows effective inference even with minimal domain knowledge, and we further introduce a formulation of Bayesian Monte Carlo to permit the principled management of the hyperparameters introduced by our flexible models. We demonstrate how our methods can be applied even in the presence of various problematic data features, including cases where our data is delayed, intermittently missing, censored and/or correlated, and we show how a decision theoretic method can be used to determine the optimal selection of observations in order to maintain a desired level of accuracy in our inference. We validate our approach using data collected from two networks of weather sensors and show that it yields better inference performance than both conventional independent Gaussian processes and the Kalman filter. Finally, we analyse the computational complexity of our algorithm.

Categories and Subject Descriptors: C.2.4 [**Computer Communication Networks**]: Distributed Systems – Distributed Applications

General Terms: Algorithms, Design, Performance

Additional Key Words and Phrases: Learning of models from data, Gaussian processes, information processing, adaptive sampling

---

[1]A preliminary version of the work presented in this paper appeared as: Osborne, M. A., Rogers, A., Ramchurn, S., Roberts, S. J. and Jennings, N. R. (2008) 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), April 2008, St. Louis, Missouri, USA. pp. 109-120.

1. INTRODUCTION

Sensor networks have recently generated a great deal of research interest within the computer and physical sciences, and their use for the scientific monitoring of remote and hostile environments is increasingly common-place. While early sensor networks were a simple evolution of existing automated data loggers that collected data for later off-line scientific analysis, more recent sensor networks now make their data available in real-time through the internet, and increasingly perform some form of real-time data processing or aggregation to provide useful summary information to users (see [Hart and Martinez 2006] for a review of such environmental sensor networks).

Providing real-time access to sensor data in this way presents many novel challenges; not least the need for self-describing data formats, and standard protocols such that the existence and capabilities of sensors can advertised to potential users. However, more significantly for us, many of the information processing tasks that would previously have been performed off-line by the expert owner or single user of an environmental sensor network (such as detecting faulty sensors, performing 'data cleaning' to remove erroneous readings, fusing noisy measurements from several sensors, and deciding how frequently data should be collected), must now be performed autonomously in real-time. Furthermore, since such information processing is likely to be performed within centralised sensor repositories (of which the Weather Underground – `http://www.wunderground.com` – and Microsoft Research's SensorMap – `http://atom.research.microsoft.com/sensormap/` – are embryonic examples), and will be applied to open sensor networks where additional sensors may be deployed at any time, and existing sensors may be removed, repositioned or updated after deployment (such as the rooftop weather sensors within the Weather Underground), these information processing tasks may have to be performed with only limited knowledge of the precise location, reliability, and accuracy of each sensor.

Now, many of the information processing tasks described above have previously been tackled by applying principled Bayesian methodologies from the academic literature of geospatial statistics and machine learning: specifically, kriging [Cressie 1991] and Gaussian processes [Rasmussen and Williams 2006]. However, due to the computational complexity of these approaches, to date they have largely been used off-line in order to analyse and re-design existing sensor networks (e.g. to reduce maintenance costs by removing the least informative sensors from an existing sensor network [Fuentes et al. 2007], or to find the optimum placement of a small number of sensors, after a trial deployment of a larger number has collected data indicating their spatial correlation [Krause et al. 2006]). Alternatively, they have been applied to single sensors and have ignored the correlations that exist between different sensors within the network [Kho et al. 2009; Padhy et al. 2010]. Thus, there is a clear need for more computationally efficient algorithms, in order that this information processing can be performed at scale in real-time.

Against this background, this paper describes our work developing just such an algorithm. More specifically, we present a novel iterative formulation of a multi-output Gaussian process (GP) that uses a computationally efficient implementation of *Bayesian Monte Carlo* to marginalise hyperparameters, efficiently re-uses previ-

ous computations by following an online update procedure as new data sequentially arrives, and uses a principled 'windowing' of data in order to maintain a reasonably sized data set. We use this GP to build a probabilistic model of the environmental variables being measured by sensors within the network, that is tolerant to data that may be missing, delayed, censored and/or correlated. This model allows us to then perform information processing tasks including: modelling the accuracy of the sensor readings, predicting the value of missing sensor readings, predicting how the monitored environmental variables will evolve in the near future, and performing active sampling by deciding when and from which sensor to acquire readings. We validate our multi-output Gaussian process formulation using data from two networks of weather sensors deployed on the south coast of England, and above and around the Swiss town of Davos, and we demonstrate its effectiveness by benchmarking it against conventional single-output Gaussian processes that model each sensor independently. Our results on this data set are promising, and represent a step towards the deployment of real-time algorithms that use principled machine learning techniques to autonomously acquire and process data from sensor networks.

The remainder of this paper is organised as follows: Section 2 describes the information processing problem that we face. Section 3 presents our Gaussian process formulation, and section 4 describes the two sensor networks used to validate it. In section 5 we present experimental results using data from these networks, and in section 6 we present results on the computational cost of our algorithm. Finally, related work is discussed in section 7, and we conclude in section 8.

2. THE INFORMATION PROCESSING PROBLEM

As discussed above, we require that our algorithm be able to autonomously perform data acquisition and information processing despite having only limited specific knowledge of each of the sensors in its local neighbourhood (e.g. their precise location, reliability, and accuracy). To this end, we require that it explicitly represents:

(1) The uncertainty in the estimated values of environmental variables being measured, noting that sensor readings will always incorporate some degree of measurement noise.
(2) The correlations or delays that exist between sensor readings; sensors that are close to one another, or in similar environments, will tend to make similar readings, while many physical processes involving moving fields (such as the movement of weather fronts) will induce delays between sensors.

We then require that it uses this representation in order to:

(1) Perform regression and prediction of environmental variables; that is, interpolate between sensor readings to predict variables at missing sensors (i.e. sensors that have failed or are unavailable through network outages), and perform short term prediction in order to support decision making.
(2) Perform efficient active sampling by selecting when to take a reading, and which sensor to read from, such that the minimum number of sensor readings are used to maintain the estimated uncertainty in environmental variables below a specified threshold (or similarly, to minimise uncertainty given a constrained

number of sensor readings). Such constraints may reflect the computational limitations of the processor on which the algorithm is running, or alternatively, where the algorithm is actually controlling the sensors within the network, it may reflect the constrained power consumption of the sensors themselves.

More specifically, the problem that we face can be cast as a multivariate regression and decision problem in which we have $l = 1 \dots L$ environmental variables $x_l \in \mathbb{R}$ of interest (such as air temperature, wind speed or direction specified at different sensor locations). We assume a set of N potentially noisy sensor readings, $\left\{ \left[[l_1, t_1], z_1 \right], \dots, \left[[l_N, t_N], z_N \right] \right\}$, in which we, for example, observe the value z_1 for the ${l_1}^{\text{th}}$ variable at time t_1, whose true unknown value is y_1. Where convenient, we may group the inputs as $x = [l, t]$. Note that we do not require that all the variables are observed at the same time, nor do we impose any discretisation of our observations into regularly spaced time steps. We define our vector of observations as $\boldsymbol{z}_d \triangleq [z_1, \dots, z_N]$ of variables labelled by $\boldsymbol{l}_d \triangleq [l_1, \dots, l_N]$ at times $\boldsymbol{t}_d \triangleq [t_1, \dots, t_N]$. Given this data, we are interested in inferring the vector of values $\boldsymbol{y}_\star$ for any other vector of variables labelled by $\boldsymbol{l}_\star$ at times $\boldsymbol{t}_\star$.

3. GAUSSIAN PROCESSES

Multivariate regression problems of the form described above are increasingly being addressed using Gaussian processes (GPs). These represent a powerful way to perform Bayesian inference about functions; we consider our environmental variables as just such a function [Rasmussen and Williams 2006]. This function takes as inputs the variable label and time pair x and produces as output the variable's value y. In this work, we will assume that our inputs are always known (e.g. our data is time-stamped), and will incorporate them into our background knowledge, or context, I. A GP is then a generalised multivariate Gaussian prior distribution over the (potentially infinite number of) outputs of this function:

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

It is specified by prior mean and covariance functions, which generate $\boldsymbol{\mu}$ and $\mathbf{K}$. The multivariate Gaussian distribution is qualified for this role due to the fact that both its marginal probabilities and conditional probabilities are themselves Gaussian. This allows us to produce analytic posterior distributions for outputs of interest, conditioned on whatever sensor readings have been observed. Furthermore, this posterior distribution will have both a predictive mean and a variance to explicitly represent our uncertainty.

While the fundamental theory of GPs is well established (see Rasmussen and Williams [2006] for example), there is much scope for the development of computationally efficient implementations especially for applications such as sensor networks which involve large numbers of sensors collecting large quantities of data timestamped data. To this end, in this work we present a novel on-line formalism of a multi-dimensional GP that allows us to model the correlations between sensor readings, and to update this model on-line as new observations are sequentially available. In the next sections we describe the covariance functions that we use to represent correlations and delays between sensor readings, the *Bayesian Monte*

Carlo method that we use to marginalise the hyperparameters of these covariance functions, and how we efficiently update the model as new data is received, by reusing the results of previous computations, and applying a principled 'windowing' of our data series.

3.1 Covariance Functions

The prior mean of a GP represents whatever we expect for our function before seeing any data. We take this as a function constant in time, such that $\mu([l,t]) = \mu_l$. The *covariance function* of a GP specifies the correlation between any pair of outputs. This can then be used to generate a covariance matrix over our set of observations and predictants. Fortunately, there exist a wide variety of functions that can serve in this purpose [Abrahamsen 1997; Stein 2005], which can then be combined and modified in a further multitude of ways. This gives us a great deal of flexibility in our modelling of functions, with covariance functions available to model periodicity, delay, noise and long-term drifts.

As an example, consider a covariance given by the Hadamard product of a covariance function over time alone and a covariance function over environmental variable labels alone, such that

$$K([l,t],[l',t']) \triangleq K_{\text{label}}(l,l')\, K_{\text{time}}(t-d_l, t'-d_{l'})\,, \tag{2}$$

where $\boldsymbol{d}$ allows us to express the delays between environmental variables. We will often use the completely general *spherical parameterisation*, $\mathbf{s}$, such that

$$K_{\text{label}}(l,l') \triangleq \operatorname{diag}(\boldsymbol{g})\, \mathbf{s}^{\mathrm{T}}\mathbf{s}\, \operatorname{diag}(\boldsymbol{g})\,, \tag{3}$$

where $\boldsymbol{g}$ gives an intuitive length scale for each environmental variable, and $\mathbf{s}^{\mathrm{T}}\mathbf{s}$ is the correlation matrix [Pinheiro and Bates 1996]. This allows us to represent any possible degree of correlation between our variables. As a less general alternative, we might instead use a term dependent on the spatial separation between sensors.

Similarly, we can represent correlations over time with a wide variety of covariance functions, permitting the incorporation of what domain knowledge we have. For example, we use the additive combination of a periodic term and a non-periodic disturbance term

$$K_{\text{time}}(t,t') \triangleq K_{\text{time},1}(t,t') + K_{\text{time},2}(t,t') \tag{4}$$

where we expect our variable to be well-represented by the superposition of a periodic and a non-periodic component. We represent both terms using the Matérn class with $\nu = 5/2$ — a covariance function that has been shown to fit a wide range of real world data streams [Rasmussen and Williams 2006] — given by

$$K_{\text{time},i}(t,t') \triangleq h^2 \left(1 + \sqrt{5}r_i + \frac{5r_i^2}{3}\right) \exp\left(-\sqrt{5}r_i\right)\,, \tag{5}$$

where $r_1 = \sin\pi\left|\frac{t-t'}{w}\right|$ for periodic terms, and $r_2 = \left|\frac{t-t'}{w}\right|$ for non-periodic terms. The Matérn class allows us to empirically select a degree of smoothness, given by the choice of ν, appropriate for the functions we are trying to track. Finally, to represent measurement noise, we further extend the covariance function to

$$V([l,t],[l',t']) \triangleq K([l,t],[l',t']) + \sigma^2\, \delta\big([l,t]-[l',t']\big)\,, \tag{6}$$

where $\delta(-)$ is the Kronecker delta and σ^2 represents the variance of additive Gaussian noise.

This choice of covariance is intended to model correlated periodic variables subject to local disturbances which may themselves be correlated amongst variables. This general model describes many environmental variables that are subject to some daily cycle (e.g. the 12 hour cycle of the tide, or the 24 hour cycle seen in most temperature readings), but we reiterate that, given different domain knowledge, a variety of other covariance functions can be chosen. For example, a more suitable covariance for air temperature was found to include an additional additive covariance term over time. This allows for the possibility of both long-term drifts in temperature occurring over the course of a week, as well as more high-frequency, hourly changes. In general, increasing the complexity of the covariance function allows us to incorporate more specific domain knowledge which will result in the Gaussian processes better fitting the observed data. However, as we shall show later, the larger number of hyperparameters that typically results from making the covariance function more complex increases the computational cost of the algorithm.

Given these examples of the plethora of possibilities for adding terms to our covariance, a natural question is when to stop. The answer is, in principle, never. The 'Occam's razor' action of Bayesian inference [MacKay 2002] will automatically lead us to select the simplest sub-model that still explains the data. Note this is true even if our prior is flat over the model complexity.

In practice, however, the flexibility of our model comes at the cost of the introduction of a number of hyperparameters, which we collectively denote as ϕ. These include correlation hyperparameters (i.e. $\boldsymbol{g}$, $\mathbf{s}$ and $\boldsymbol{d}$), along with others such as the periods and amplitudes of each covariance term (i.e. w and h) and the noise deviation σ. The constant prior means $\mu_1, \ldots, \mu_M$ are also included as additional hyperparameters. Taking these hyperparameters as given and using the properties of the Gaussian distribution, we are able to write our predictive equations as

$$p(\, \boldsymbol{x}_\star \mid \boldsymbol{y}_d,\, \phi,\, I\,) = \mathrm{N}(\boldsymbol{x}_\star; \boldsymbol{m}_*, \mathbf{C}_*)\ , \tag{7}$$

where, collecting our inputs as $\boldsymbol{x}_\star \triangleq [\boldsymbol{l}_\star, \boldsymbol{t}_\star]$ and $\boldsymbol{x}_d \triangleq [\boldsymbol{l}_d, \boldsymbol{t}_d]$, we have:

$$\boldsymbol{m}_* = \boldsymbol{\mu}_\phi(\boldsymbol{x}_\star) + \mathbf{K}_\phi(\boldsymbol{x}_\star, \boldsymbol{x}_d)\mathbf{V}_\phi(\boldsymbol{x}_d, \boldsymbol{x}_d)^{-1}(\boldsymbol{y}_d - \boldsymbol{\mu}_\phi(\boldsymbol{x}_d)) \tag{8}$$

$$\mathbf{C}_* = \mathbf{K}_\phi(\boldsymbol{x}_\star, \boldsymbol{x}_\star) - \mathbf{K}_\phi(\boldsymbol{x}_\star, \boldsymbol{x}_d)\mathbf{V}_\phi(\boldsymbol{x}_d, \boldsymbol{x}_d)^{-1}\mathbf{K}_\phi(\boldsymbol{x}_d, \boldsymbol{x}_\star) \tag{9}$$

3.2 Marginalisation

Of course, it is rare that we can be certain a priori about the values of our hyperparameters. Rather than equation (7), therefore, we must consider

$$p(\, \boldsymbol{y}_\star \mid \boldsymbol{z}_d, I\,) = \frac{\int \mathrm{d}\phi\, p(\, \boldsymbol{y}_\star \mid \boldsymbol{z}_d, \phi, I\,)\, p(\, \boldsymbol{z}_d \mid \phi, I\,)\, p(\, \phi \mid I\,)}{\int \mathrm{d}\phi\, p(\, \boldsymbol{z}_d \mid \phi, I\,)\, p(\, \phi \mid I\,)}\,, \tag{10}$$

in which we have marginalised ϕ. Unfortunately, both our likelihood $p(\, \boldsymbol{z}_d \mid \phi, I\,)$ and predictions $p(\, \boldsymbol{y}_\star \mid \boldsymbol{z}_d, \phi, I\,)$ exhibit non-trivial dependence upon ϕ and so our integrals are non-analytic. As such, we resort to quadrature, which inevitably

involves evaluating the two quantities

$$\begin{aligned} q(\phi) &\triangleq p(\, \boldsymbol{y}_\star \mid \boldsymbol{z}_d,\, \phi,\, I\,) \\ r(\phi) &\triangleq p(\, \boldsymbol{z}_d \mid \phi,\, I\,) \end{aligned} \tag{11}$$

at a set of sample points $\boldsymbol{\phi}_s = \{\phi_1, \ldots, \phi_\eta\}$, giving $\boldsymbol{q}_s \triangleq q(\boldsymbol{\phi}_s)$ and $\boldsymbol{r}_s \triangleq r(\boldsymbol{\phi}_s)$. Of course, this evaluation is a computationally expensive operation. Defining the vector of all possible function inputs as $\boldsymbol{\phi}$, we clearly can't afford to evaluate $\boldsymbol{q} \triangleq q(\boldsymbol{\phi})$ or $\boldsymbol{r} \triangleq r(\boldsymbol{\phi})$. Note that the more complex our model, and hence the greater the number of hyperparameters, the higher the dimension of the hyperparameter space we must sample in. As such, the complexity of models we can practically consider is limited by the curse of dimensionality. We can view our sparse sampling as introducing a form of uncertainty about the functions q and r, which we can again address using Bayesian probability theory.

To this end, we apply *Bayesian Monte Carlo*, and thus, assign further GP priors to q and r [Rasmussen and Ghahramani 2003]. This choice is motivated by the fact that variables over which we have a multivariate Gaussian distribution are joint Gaussian with any projections of those variables. As such, given that integration is a projection, we can use our computed samples $\boldsymbol{q}_s$ in order to perform Gaussian process regression about the value of integrals over $q(\phi)$, and similarly for r. Note that the quantity we wish to perform inference about,

$$\psi \triangleq p(\, \boldsymbol{y}_\star \mid \boldsymbol{q},\, \boldsymbol{r},\, \boldsymbol{z}_d, I\,) = \frac{\int q(\phi_\star)\, r(\phi_\star)\, p(\, \phi_\star \mid I\,)\, \mathrm{d}\phi_\star}{\int r(\phi_\star)\, p(\, \phi_\star \mid I\,)\, \mathrm{d}\phi_\star}\,, \tag{12}$$

possesses richer structure than that previously considered using Bayesian Monte Carlo techniques. In our case, $r(\phi)$ appears in both our numerator and denominator integrals, introducing correlations between the values we estimate for them. The correlation structure of this system is illustrated in Figure 1.

In considering any problem of inference, we need to be clear about both what information we have and which uncertain variables we are interested in. In our case, both function values, $\boldsymbol{q}_s$ and $\boldsymbol{r}_s$, and their locations, $\boldsymbol{\phi}_s$, represent valuable pieces of knowledge[2]. As with our convention above, we will take knowledge of sample locations $\boldsymbol{\phi}_s$ to be implicit within I. We respectively define $\boldsymbol{m}^{(q)}$ and $\boldsymbol{m}^{(r)}$ as the means for $\boldsymbol{q}$ and $\boldsymbol{r}$ conditioned on $\boldsymbol{q}_s$ and $\boldsymbol{r}_s$ from (8), $\mathbf{C}^{(r)}$ and $\mathbf{C}^{(r)}$ the similarly conditional covariances from (9). The ultimate quantity of our interest is then

$$\begin{aligned} &p(\, \boldsymbol{y}_\star \mid \boldsymbol{q}_S,\, \boldsymbol{r}_S,\, \boldsymbol{z}_d,\, I\,) \\ &= \iiint p(\, \boldsymbol{y}_\star \mid \boldsymbol{q},\, \boldsymbol{r},\, \boldsymbol{z}_d, I\,)\; p(\, \psi \mid \boldsymbol{q},\, \boldsymbol{r},\, I\,)\; p(\, \boldsymbol{q} \mid \boldsymbol{q}_S,\, I\,)\; p(\, \boldsymbol{r} \mid \boldsymbol{r}_S,\, I\,)\; \mathrm{d}\psi\, \mathrm{d}\boldsymbol{q}\, \mathrm{d}\boldsymbol{r} \\ &= \iiint \psi\, \delta\!\left(\psi - \frac{\int q_*\, r_*\, p(\, \phi_\star \mid I\,)\, \mathrm{d}\phi_\star}{\int r_*\, p(\, \phi_\star \mid I\,)\, \mathrm{d}\phi_\star}\right) \mathrm{N}\!\left(\boldsymbol{q}; \boldsymbol{m}^{(q)}, \mathbf{C}^{(q)}\right) \mathrm{N}\!\left(\boldsymbol{r}; \boldsymbol{m}^{(r)}, \mathbf{C}^{(r)}\right) \mathrm{d}\psi\, \mathrm{d}\boldsymbol{q}\, \mathrm{d}\boldsymbol{r} \\ &= \int \frac{\int m_*^{(q)}\, r_*\, p(\, \phi_\star \mid I\,)\, \mathrm{d}\phi_\star}{\int r_*\, p(\, \phi_\star \mid I\,)\, \mathrm{d}\phi_\star} \mathrm{N}\!\left(\boldsymbol{r}; \boldsymbol{m}^{(r)}, \mathbf{C}^{(r)}\right) \mathrm{d}\boldsymbol{r}\,. \end{aligned} \tag{13}$$

[2]As discussed by [O'Hagan 1987], traditional, frequentist Monte Carlo effectively ignores the information content of $\boldsymbol{\phi}_s$, leading to several unsatisfactory features.

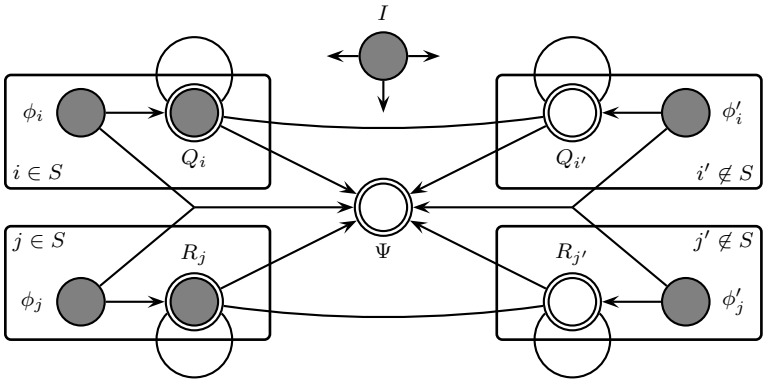

Fig. 1. Bayesian network for marginalising hyperparameters using Bayesian Monte Carlo. Shaded nodes are known and double-circled nodes are deterministic given all their parents. All Q nodes are correlated with one another, as are all R nodes, and the context I is correlated with all nodes.

Here, unfortunately, our integration over r becomes nonanalytic. However, we can employ a Laplace approximation by expanding the integrand around an assumed peak at $\boldsymbol{m}^{(r)}$. Before we can state its result, to each of our hyperparameters we assign a Gaussian prior distribution (or if our hyperparameter is restricted to the positive reals, we instead assign a Gaussian distribution to its log) given by

$$p(\,\phi \mid I\,) \triangleq \mathrm{N}\big(\phi; \boldsymbol{\nu}, \lambda^{\mathrm{T}}\lambda\big)\ . \tag{14}$$

Note that the intuitive spherical parameterisation (3) assists with the elicitation of priors over its hyperparameters. We then assign a *squared exponential* covariance function for the GP over both q and r given by

$$K(\phi, \phi') \triangleq \mathrm{N}\big(\phi; \phi', \mathbf{w}^{\mathrm{T}}\mathbf{w}\big)\ . \tag{15}$$

Finally, using the further definition for $i, j \in \mathcal{I}_s$, that

$$\mathfrak{N}_s(i,j) \triangleq \mathrm{N}\left(\begin{bmatrix}\boldsymbol{\phi}_i\\ \boldsymbol{\phi}_j\end{bmatrix}; \begin{bmatrix}\boldsymbol{\nu}\\ \boldsymbol{\nu}\end{bmatrix}, \begin{bmatrix}\lambda^{\mathrm{T}}\lambda + \mathbf{w}^{\mathrm{T}}\mathbf{w} & \lambda^{\mathrm{T}}\lambda\\ \lambda^{\mathrm{T}}\lambda & \lambda^{\mathrm{T}}\lambda + \mathbf{w}^{\mathrm{T}}\mathbf{w}\end{bmatrix}\right)\ , \tag{16}$$

our Laplace approximation gives us

$$p(\,\boldsymbol{y}_\star \mid \boldsymbol{y}_d, I\,) \simeq \boldsymbol{q}_s^{\mathrm{T}}\,\boldsymbol{\rho}\,, \tag{17}$$

where the weights of this linear combination are

$$\boldsymbol{\rho} \triangleq \frac{\mathbf{K}(\boldsymbol{\phi}_s, \boldsymbol{\phi}_s)^{-1}\,\mathfrak{N}_s\,\mathbf{K}(\boldsymbol{\phi}_s, \boldsymbol{\phi}_s)^{-1}\,\boldsymbol{r}_s}{\mathbf{1}_{s,1}^{\mathrm{T}}\,\mathbf{K}(\boldsymbol{\phi}_s, \boldsymbol{\phi}_s)^{-1}\,\mathfrak{N}_s\,\mathbf{K}(\boldsymbol{\phi}_s, \boldsymbol{\phi}_s)^{-1}\,\boldsymbol{r}_s}\,, \tag{18}$$

and $\mathbf{1}_{s,1}$ is a vector containing only ones of dimensions equal to $\boldsymbol{q}_s$. With a GP on $p(\,\boldsymbol{y}_\star \mid \phi, I\,)$, each $q_i = p(\,\boldsymbol{y}_\star \mid \boldsymbol{z}_d, \phi_i, I\,)$ will be a slightly different Gaussian. Hence we effectively approximate $p(\,\boldsymbol{y}_\star \mid \boldsymbol{z}_d, I\,)$ as a Gaussian (process) mixture; Bayesian Monte Carlo returns a weighted sum of our predictions evaluated at a sample set of hyperparameters. The assignment of these weights is informed by the best use of all pertinent information. As such, it avoids the risk of overfitting that occurs when applying a less principled technique such as likelihood maximisation [MacKay 2002].

3.3 Censored Observations

In the work above, we have assumed that observations of our variables of interest were corrupted by simple Gaussian noise. However, in many contexts, we instead observe *censored* observations. That is, we might observe that a variable was above or below certain thresholds, but no more. Examples are rich within the weather sensor networks considered. Float sensors are prone to becoming lodged on sensor posts, reporting only that the water level is below that at which it is stuck. Other observations are problematically rounded to the nearest integer – if we observe a reading of x, we can say only that the true value was between $x - 0.5$ and $x + 0.5$. We can readily extend our inference framework to allow for such a noise model.

More precisely, we assume that we actually observe bounds $\boldsymbol{b}_c$ that constrain Gaussian-noise corrupted versions $\boldsymbol{z}_c$ of the underlying variables of interest $\boldsymbol{y}_c$ at $\boldsymbol{x}_c$. This framework allows for imprecise censored observations. Note that the noise variance for censored observations may differ from the noise variance associated with other observations. Conditioned on a combination of censored and un-censored observations, the distribution for our variables of interest is

$$p(\,\boldsymbol{y}_\star \mid \boldsymbol{z}_d, \boldsymbol{b}_c, I\,) = \frac{\int \mathrm{d}\phi \int_{\boldsymbol{b}_c} \mathrm{d}\boldsymbol{z}_c\, p(\,\boldsymbol{y}_\star \mid \boldsymbol{z}_d, \boldsymbol{z}_c, \phi, I\,)\, p(\,\boldsymbol{z}_c \mid \boldsymbol{z}_d, \phi, I\,)\, p(\,\boldsymbol{z}_d \mid \phi, I\,)\, p(\,\phi \mid I\,)}{\int \mathrm{d}\phi \int_{\boldsymbol{b}_c} \mathrm{d}\boldsymbol{z}_c\, p(\,\boldsymbol{z}_c \mid \boldsymbol{z}_d, \phi, I\,)\, p(\,\boldsymbol{z}_d \mid \phi, I\,)\, p(\,\phi \mid I\,)}\,. \tag{19}$$

While we cannot determine this full, non-Gaussian distribution easily, we can analytically determine its mean and covariance. We use the abbreviations $\boldsymbol{m}_{c|d} \triangleq \boldsymbol{m}(\boldsymbol{z}_c \,|\, \boldsymbol{z}_d, \phi, I)$ and $\mathbf{C}_{c|d} \triangleq \mathbf{C}(\boldsymbol{z}_c \,|\, \boldsymbol{z}_d, \phi, I)$. To reflect the influence of our censored observations, the first required modification to our previous results is to incorporate a new term into our likelihoods,

$$r^{(cd)}(\phi) = \mathrm{N}(\boldsymbol{z}_d; \boldsymbol{m}(\boldsymbol{z}_d \,|\, \phi, I)\,, \mathbf{C}(\boldsymbol{z}_d \,|\, \phi, I)) \int_{\boldsymbol{b}_c} \mathrm{d}\boldsymbol{z}_c\, \mathrm{N}\big(\boldsymbol{z}_c; \boldsymbol{m}_{c|d}, \mathbf{C}_{c|d}\big)\,, \tag{20}$$

giving the new weights over hyperparameter samples

$$\boldsymbol{\rho}^{(cd)} \triangleq \frac{\mathbf{K}(\boldsymbol{\phi}_s, \boldsymbol{\phi}_s)^{-1}\, \mathfrak{N}_s\, \mathbf{K}(\boldsymbol{\phi}_s, \boldsymbol{\phi}_s)^{-1}\, \boldsymbol{r}_s^{(cd)}}{\mathbf{1}_{s,1}^{\mathrm{T}}\, \mathbf{K}(\boldsymbol{\phi}_s, \boldsymbol{\phi}_s)^{-1}\, \mathfrak{N}_s\, \mathbf{K}(\boldsymbol{\phi}_s, \boldsymbol{\phi}_s)^{-1}\, \boldsymbol{r}_s^{(cd)}}\,. \tag{21}$$

We can then write our predictive mean as

$$\boldsymbol{m}(\,\boldsymbol{y}_\star \mid \boldsymbol{z}_d, \boldsymbol{b}_c, I\,) = \sum_{i \in s} \rho_i^{(cd)} \Bigg(\boldsymbol{\mu}_{\phi_i}(\boldsymbol{x}_\star) + \mathbf{K}_{\phi_i}(\boldsymbol{x}_\star, [\boldsymbol{x}_c, \boldsymbol{x}_d]) \mathbf{V}_{\phi_i}([\boldsymbol{x}_c, \boldsymbol{x}_d], [\boldsymbol{x}_c, \boldsymbol{x}_d])^{-1} \begin{bmatrix} \frac{\int_{\boldsymbol{b}_c} \mathrm{d}\boldsymbol{z}_c\, \boldsymbol{z}_c\, \mathrm{N}\left(\boldsymbol{z}_c; \boldsymbol{m}_{c|d}, \mathbf{C}_{c|d}\right)}{\int_{\boldsymbol{b}_c} \mathrm{d}\boldsymbol{z}_c\, \mathrm{N}\left(\boldsymbol{z}_c; \boldsymbol{m}_{c|d}, \mathbf{C}_{c|d}\right)} - \boldsymbol{\mu}_{\phi_i}(\boldsymbol{x}_c) \\ \boldsymbol{y}_d - \boldsymbol{\mu}_{\phi_i}(\boldsymbol{x}_d) \end{bmatrix} \Bigg), \tag{22}$$

noting that a censored observation is intuitively treated as an uncensored observation equal to the conditional mean of the GP over the bounded region. We have

also the predictive covariance

$$\mathbf{C}(\,\boldsymbol{y}_\star \mid \boldsymbol{z}_d, \boldsymbol{b}_c, I\,) = \sum_{i\in s} \rho_i^{(cd)} \Big(\mathbf{K}_{\phi_i}(\boldsymbol{x}_\star, \boldsymbol{x}_\star) - \mathbf{K}_{\phi_i}(\boldsymbol{x}_\star, [\boldsymbol{x}_c, \boldsymbol{x}_d]) \mathbf{V}_{\phi_i}([\boldsymbol{x}_c, \boldsymbol{x}_d], [\boldsymbol{x}_c, \boldsymbol{x}_d])^{-1} \mathbf{K}_{\phi_i}([\boldsymbol{x}_c, \boldsymbol{x}_d], \boldsymbol{x}_\star) \Big) . \quad (23)$$

We now have the problem of approximating the integrals $\int_{\boldsymbol{b}_c} \mathrm{d}\boldsymbol{z}_c \, \mathrm{N}\big(\boldsymbol{z}_c; \boldsymbol{m}_{c|d}, \mathbf{C}_{c|d}\big)$ and $\int_{\boldsymbol{b}_c} \mathrm{d}\boldsymbol{z}_c \, \boldsymbol{z}_c \, \mathrm{N}\big(\boldsymbol{z}_c; \boldsymbol{m}_{c|d}, \mathbf{C}_{c|d}\big)$, which are non-analytic. Fortunately, there exists an efficient Monte Carlo algorithm [Genz 1992] for exactly this purpose.

3.4 Efficient Implementation

Now the implementation of equations (8) and (9) involves the inverse of a matrix, and the most stable way to implement this is through the use of the Cholesky decomposition, $\mathbf{R}(\boldsymbol{x}_d, \boldsymbol{x}_d)$, of $\mathbf{V}(\boldsymbol{x}_d, \boldsymbol{x}_d)$, rather than inverting the matrix directly. Performing this Cholesky decomposition represents the most computationally expensive operation we must perform; its cost scaling as $O(N^3)$ in the number of data points N. However, as discussed earlier, we do not intend to use our GP with a fixed set of data, but rather, within an on-line algorithm that receives new observations over time. As such, we must be able to iteratively update our predictions in as little time as possible. Fortunately, we can do so by exploiting the special structure of our problem. When we receive new data, our $\mathbf{V}$ matrix is changed only in the addition of a few new rows and columns. Hence most of the work that went into computing its Cholesky decomposition at the last iteration can be recycled to produce the new Cholesky decomposition (see Appendix A.1 for details of this operation). Another problematic calculation required by (8) and (9) is the computation of the data-dependent term $\mathbf{R}(\boldsymbol{x}_d, \boldsymbol{x}_d)^{-1}(\boldsymbol{y}_d - \boldsymbol{\mu}(\boldsymbol{x}_d))$, in which $\boldsymbol{y}_d - \boldsymbol{\mu}(\boldsymbol{x}_d)$ is also only changing due to the addition of new rows. As such, efficient updating rules are also available for this term (see Appendix A.2). As such, we are able to reduce the overall cost of an update from $O(N^3)$ to $O(N^2)$.

However, we can further increase the efficiency of our updates by making a judicious assumption. In particular, experience shows that our GP requires only a very small number of recent observations in order to produce good estimates. Indeed, most covariance functions have very light tails such that only points within a few multiples of the time scale are at all relevant to the point of interest. Hence we seek sensible ways of discarding information once it has been rendered 'stale', to reduce both memory usage and computational requirements.

One pre-eminently reasonable measure of the value of data is the uncertainty we still possess after learning it. In particular, we are interested in how uncertain we are about $\boldsymbol{x}_\star$; as given by the covariance of our Gaussian mixture equation (17). Our approach is thus to drop our oldest data points (those which our covariance deems least relevant to the current predictant) until this uncertainty exceeds some predetermined threshold.

Just as we were able to efficiently update our Cholesky factor upon the receipt of new data, so we can downdate to remove data (see Appendix A.3 for the details of this operation). This allows us to rapidly remove unwanted data, compute our uncertainty about $\boldsymbol{y}_\star$, and then repeat as required; the GP will retain only as

much data as necessary to achieve a pre-specified degree of accuracy. This allows a principled way of 'windowing' our data series.

Finally, we turn to the implementation of our marginalisation procedure. Essentially, our approach is to maintain an ensemble of GPs, one for each hyperparameter sample, running in parallel, each of which we update and downdate according to the proposals above. Their predictions are then weighted and combined according to equation (17). Note that the only computations whose computational cost grows at greater than a quadratic rate in the number of samples, η, are the Cholesky decomposition and multiplication of covariance matrices in equation (17), and these scale rather poorly as $O(\eta^3)$. To address this problem, we take our Gaussian priors for each different hyperparameter $\phi_{(e)} \in \phi$ as independent. We further take a covariance structure given by the product of terms over each hyperparameter, the common *product correlation rule* (e.g. Sasena [2002])

$$K(\phi,\, \phi') = \prod_e K_e\Big(\phi_{(e)},\, \phi'_{(e)}\Big)\,. \tag{24}$$

If we additionally consider a simple grid of samples, such that $\boldsymbol{\phi}_s$ is the tensor product of a set of samples $\boldsymbol{\phi}_{(e),S}$ over each hyperparameter, then the problematic term in equation (17) reduces to the Kronecker product of the equivalent term over each individual hyperparameter:

$$\begin{aligned}
&\mathbf{K}(\boldsymbol{\phi}_s, \boldsymbol{\phi}_s)^{-1}\, \mathfrak{N}_s\, \mathbf{K}(\boldsymbol{\phi}_s, \boldsymbol{\phi}_s)^{-1} \\
&\qquad = \mathbf{K}(\boldsymbol{\phi}_{(1),S}, \boldsymbol{\phi}_{(1),S})^{-1} \mathfrak{N}_s(\boldsymbol{\phi}_{(1),S}, \boldsymbol{\phi}_{(1),S}) \mathbf{K}(\boldsymbol{\phi}_{(1),S}, \boldsymbol{\phi}_{(1),S})^{-1} \\
&\qquad\quad \otimes \mathbf{K}(\boldsymbol{\phi}_{(2),S}, \boldsymbol{\phi}_{(2),S})^{-1} \mathfrak{N}_s(\boldsymbol{\phi}_{(2),S}, \boldsymbol{\phi}_{(2),S}) \mathbf{K}(\boldsymbol{\phi}_{(2),S}, \boldsymbol{\phi}_{(2),S})^{-1} \\
&\qquad\quad \otimes \dots
\end{aligned} \tag{25}$$

This means that we only have to perform the expensive Cholesky factorisation and multiplication with matrices whose size equals the number of samples for each hyperparameter, rather than on a matrix of size equal to the total number of hyperparameter samples. This hence represents an effective way to avoid the 'curse of dimensionality'.

Applied together, these features provide us with an efficient on-line algorithm that can be applied in real-time as data is sequentially collected from the sensor network.

3.5 Active Data Selection

Finally, in addition to the regression and prediction problem described in section 2, we are able to use the same algorithm to perform active data selection. This is a decision problem concerning which observations should be taken. In this, we once again take a utility that is a function of the uncertainty in our predictions. We specify a utility of negative infinity if our uncertainty about any variable is greater than a pre-specified threshold, and a fixed negative utility is assigned as the cost of an observation (in general, this cost could be different for different sensors). Note that the uncertainty increases monotonically in the absence of new data, and shrinks in the presence of an observation. Hence our algorithm is simply induced to make a reading whenever the uncertainty grows beyond a pre-specified threshold.

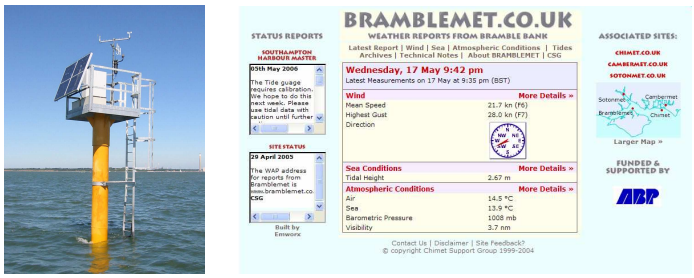

Fig. 2. The Bramble Bank weather station and web site.

Our algorithm can also decide which observation to make at this time, by determining which sensor will allow it the longest period of grace until it would be forced to observe again. This clearly minimises the number of costly observations. Note that the uncertainty of a GP, as given by (9), is actually dependent only on the location of the observation, not its actual value. Hence the uncertainty we imagine remaining after taking an observation from a sensor can be quickly determined without having to speculate about what data we might possibly collect. However, this is true only so long as we do not consider the impact of new observations on our hyperparameter sample weights (18). Our approach is to take the model, in particular the weights over samples, as fixed, and investigate only how different schedules of observations affect our predictions within it. With this proviso, we are guaranteed to maintain our uncertainty below a specified threshold, while taking as few observations as possible.

4. TRIAL IMPLEMENTATION

In order to empirically evaluate the information processing algorithm described in the previous section, we have used data from two networks of weather sensors: the *Bramblemet* weather sensor network which consists of four weather and tide sensors deployed on the south coast of England to provide real-time weather information to commercial and recreational users of the Port of Southampton (see `http://www.bramblemet.co.uk/`), and the *Wannengrat Alpine Observatory* being built above and around the town of Davos as part of Swiss Experiment (see `http://www.swiss-experiment.ch/index.php/Wannengrat:Home`). The use of such weather sensors is attractive since they exhibit challenging correlations and delays whose physical processes are well understood. Hence, the relationships learned by the Gaussian process can be compared to these processes.

4.1 Bramblemet Weather Sensor Network

In more detail, the Bramblemet network consists of four sensors (named Bramblemet, Sotonmet, Cambermet and Chimet), each of which measures a range of environmental variables (including wind speed and direction, air temperature, sea temperature, and tide height) and makes up-to-date sensor measurements available through separate web pages (see figure 2 and `http://www.bramblemet.co.uk/`). To facilitate the autonomous collection of sensor data by our information processing algorithm, we have worked with the sensor network operators (Associated British

```
<sit:Location rdf:about="&sit;bramblemet"
  rdfs:label="Bramble Bank"
  geo:lat="50.79472"
  geo:lng="-1.2875"
  sit:altitude="1">
</sit:Location>

<sit:Sensor rdf:about="&sit;bramblemet/windspeed"
  rdfs:label="Wind speed">
  <sit:sensorType rdf:resource="&sit;windspeed"/>
  <sit:location rdf:resource="&sit;bramblemet"/>
</sit:Sensor>

<sit:SensorType rdf:about="&sit;windspeed"
  rdfs:label="Wind speed">
</sit:SensorType>

<sit:Unit rdf:about="&sit;knots"
  rdfs:label="Knots"
  sit:unitAbbr="kn">
</sit:Unit>

<sit:Reading
  rdf:about="&sit;bramblemet/windspeed/reading/1234"
  rdfs:value="9.3"
  sit:datetime="2007-10-25T21:55:00">
  <sit:sensor rdf:resource="&sit;bramblemet/windspeed"/>
  <sit:unit rdf:resource="&sit;knots"/>
</sit:Reading>
```

Fig. 3. Example RDF data from the Bramblemet sensor.

Ports) and supplemented each sensor web page with machine readable RDF data (see figure 3 for an example of this format – current sensor data in this format is available at `http://www.bramblemet.co.uk/bra.rdf`). This format is attractive as it represents a fundamental element of the semantic web, and there exist a number of software tools to parse, store and query it. More importantly, it allows the sensor data to be precisely defined through standard ontologies [Lassila and Swick 1999]. For example, linking the predicate *geo:lat* to the ontology available at `http://www.w3.org/2003/01/geo/wgs84_pos#` precisely defines the value *"50.79472"* as representing a latitude in the WGS84 geodetic reference datum. While ontologies for sensor data have yet to be standardised, a number of candidates exist (see [Barnaghi et al. 2009] for an example based upon Sensor Web Enablement (SWE) and SensorML data component models).

In order to visualise the sensor data and the information processing algorithms in operation, we have implemented a server-based application that collects the RDF data from the sensors, parses and stores is using Jena (see `http://jena.sourceforge.net/`), and displays sensor data in tabular and graphical forms using a Google Maps interface (see figure 4). A live version of this system is available at `http://www.aladdinproject.org/situation/`.

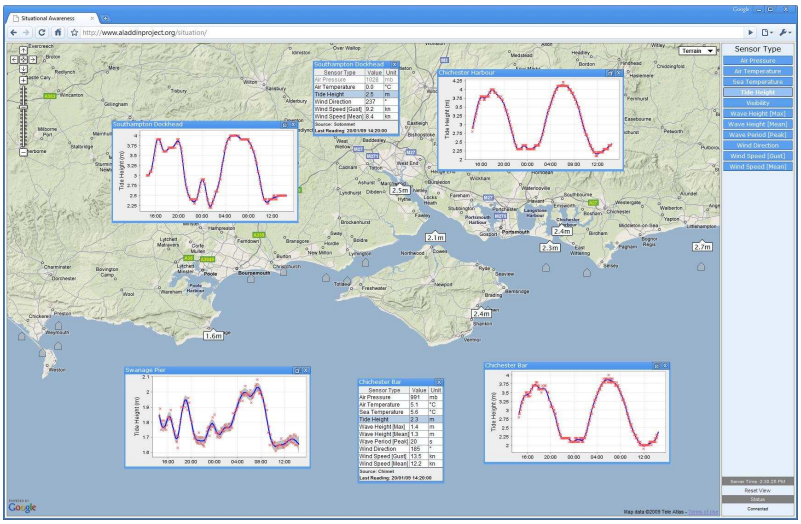

Fig. 4. Server-based implementation of our information processing algorithm accessed through a web based Google Maps interface (available at `http://www.aladdinproject.org/situation/`).

The use of the Bramblemet sensor network is particularly attractive since the network is subject to real network outages that generate instances of missing sensor readings on which we can evaluate our information processing algorithms. Furthermore, by working with the sensor network operators we have been able to recover the missing data from these periods from the sensors' local caches, and thus, perform quantitative comparisons betweeen our information processing algorithm's predictions and the actual data. Furthermore, within the Bramblemet network sensors, the sensor measurements are rounded prior to their transmission over the wireless link. This results in censored observations, and thus, again by accessing the raw data from the sensors' local caches, we are also able to compare the ability of our information processing algorithms to handle these censored observations.

However, the Bramblemet sensor network only consists of four sensors, and thus, in order to compare the performance of our approach on larger networks we turn to the Wannengrat Alpine Observatory.

4.2 Wannengrat Alpine Observatory

The Wannengrat site consists of over twenty SensorScope stations, deployed above and around the Swiss town of Davos, to measure local weather conditions (see http://www.swiss-experiment.ch/index.php/Wannengrat:Sensorscopedeployment). In the case of the Wannengrat network, we simply use the data from online data repository, and have had no direct interaction with the sensor network operators.

5. EMPIRICAL EVALUATION

In this section we empirically evaluate our information processing algorithm on real weather data collected from the sensor networks described above. We compare our multi-output GP formalism against a number of benchmarks:

Conventional independent GP in which each environmental variable is modelled separately (i.e. correlations between these parameters are ignored).

Kalman filter as a dynamic auto-regressive model, in the form of a state-space model, performing sequential predictions [Jazwinski 1970; Lee and Roberts 2008].

Naïve algorithm which simply predicts that the variable at the next time step will be equal to that most recently observed at that sensor.

The first benchmark illustrates the effectiveness of our Gaussian process formalism that expresses correlations between different sensors. The second benchmark is a state-of-the-art alternative, that has previously been used for tracking environmental sensor data [Bertino et al. 2003]. Finally, the naïve third benchmark provides a lower bound on the performance of a prediction algorithm.

In our comparison, we present results for three different sensor types: tide height, air temperature and air pressure. Tide height was chosen since it demonstrates the ability of the GP to learn and predict periodic behaviour, and more importantly, because this particular data set contains an interesting period in which extreme weather conditions (a Northerly gale) cause both an unexpectedly low tide and a failure of the wireless connection between several of the sensor and the shore that prevents our algorithm acquiring sensor readings. Air temperature was chosen since they exhibit a very different noise and correlation structure to the tide height measurements, and thus demonstrate that the generic approach described here is still able to perform reliable regression and prediction. Finally, air pressure was chosen as a demonstration of our effectiveness in processing censored observations, as in the case of the Bramblemet sensor network, the air pressure readings are subject to (the reasonably severe) rounding to the nearest Pascal.

5.1 Regression and Prediction

Figures 5 and 6 illustrate the efficacy of our GP formalism in this scenario. We plot the sensor readings acquired by our algorithm (shown as markers), the mean and standard deviation of the GP prediction (shown as a solid line with plus or minus a single standard deviation shown as shading), and the true fine-grained sensor readings (shown as bold) that were downloaded directly from the sensor (rather than through the web site) after the event. Note that we present just two sensors for reasons of space, but we use readings from all four sensors in order to perform inference. At time t, Figure 5 depicts the posterior distribution of the GP, conditioned on all observations prior to and inclusive of t. Figure 6 demonstrates our algorithm's one-step ahead predictive performance, depicting the posterior distribution at time t conditioned on all observations prior to and inclusive of $t - 5\,\text{mins}$.

We first consider figure 5 showing the tide predictions, and specifically, we note the performance of our multi-output GP formalism when the Bramblemet sensor drops out at $t = 1.45$ days. In this case, the independent GP quite reasonably predicts that the tide will repeat the same periodic signal it has observed in the past. However, the GP can achieve better results if it is allowed to benefit from the knowledge of the other sensors' readings during this interval of missing data. Thus, in the case of the multi-output GP, by $t = 1.45$ days, the GP has successfully determined that the sensors are all very strongly correlated. Hence, when it sees an unexpected low tide in the Chimet sensor data (caused by the strong Northerly

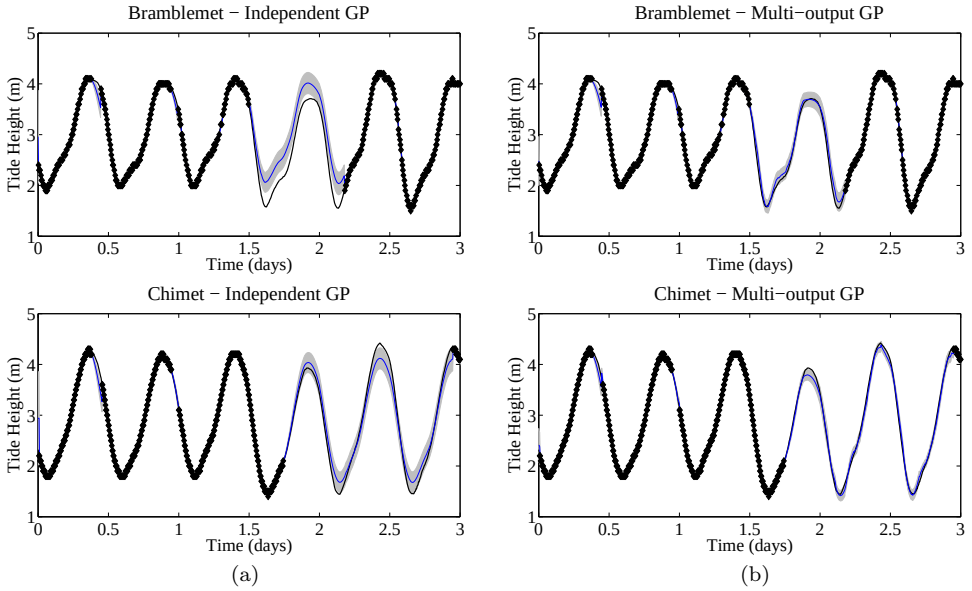

Fig. 5. Prediction and regression of tide height data for (a) independent and (b) multi-output Gaussian processes.

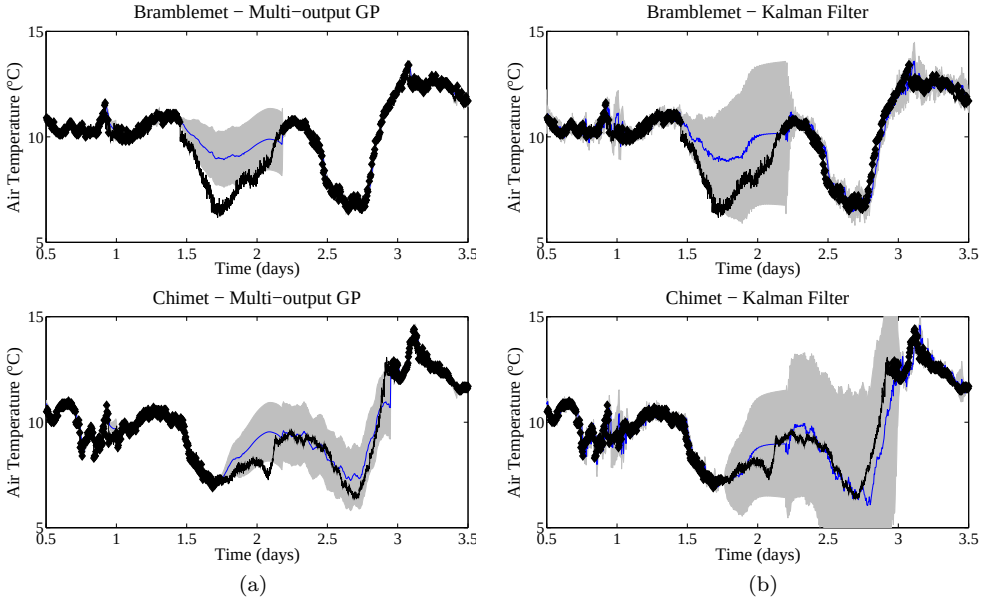

Fig. 6. One-step lookahead prediction of air temperature data for (a) a multi-output Gaussian process and (b) a Kalman filter.

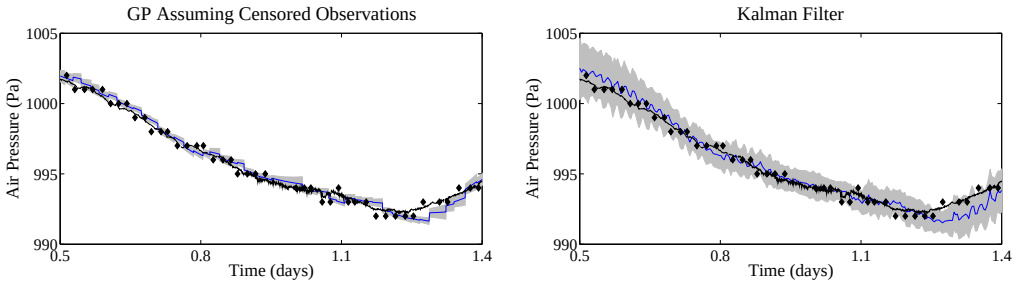

Fig. 7. Prediction and regression of air pressure data for (a) a Gaussian process employing censored observations and (b) a Kalman filter.

wind), these correlations lead it to infer a similarly low tide in the Bramblemet reading. Hence, the multi-output GP produces significantly more accurate predictions during the missing data interval, with associated smaller error bars. Exactly the same effect is seen in the later predictions of the Chimet tide height, where the multi-output GP predictions use observations from the other sensors to better predict the high tide height at $t = 2.45$ days.

Furthermore, figure 6 shows the air temperature sensor readings where a similar effect is observed. Again, the multi-output GP is able to better predict the missing air temperature readings from the Chimet sensor having learnt the correlation with other sensors, despite the fact that the data set is much noisier and the correlations between sensors are much weaker. In this, it also demonstrates a significant improvement in performance over Kalman filter predictions on the same data. The root mean square errors (RMSEs) are 0.7395 °C for our multi-output GP, 0.9159 °C for the Kalman filter and 3.8200 °C for the naïve algorithm that uses the current reading as a prediction of the next reading.

5.2 Censored Observations

Figure 7 demonstrates regression and prediction over the rounded air pressure observations from the Bramblemet sensor alone. In this case, explicitly modelling the noise process and the rounding of data separately (as described in section 3.3), as opposed to attempting to capture the rounding of data directly through the noise process (the approach adopted by the Kalman filter here), leads to a dramatic improvement in prediction performance. In this case, the RMSEs are 0.3851 Pa for the GP, 3.2900 Pa for the Kalman filter and 3.6068 Pa for the naïve algorithm. Both the GP and Kalman filter have an order of 16; that is, they store only up to the 16 most recent observations.

5.3 Active Data Selection

We now demonstrate our active data selection algorithm. Using the fine-grained data (downloaded directly from the sensors), we can simulate how our GP would have chosen its observations had it been in control. Results from the active selection of observations from all the four tide sensors are displayed in figure 8, and for three wind speed sensors in figure 9. Again, these plots depict dynamic choices; at

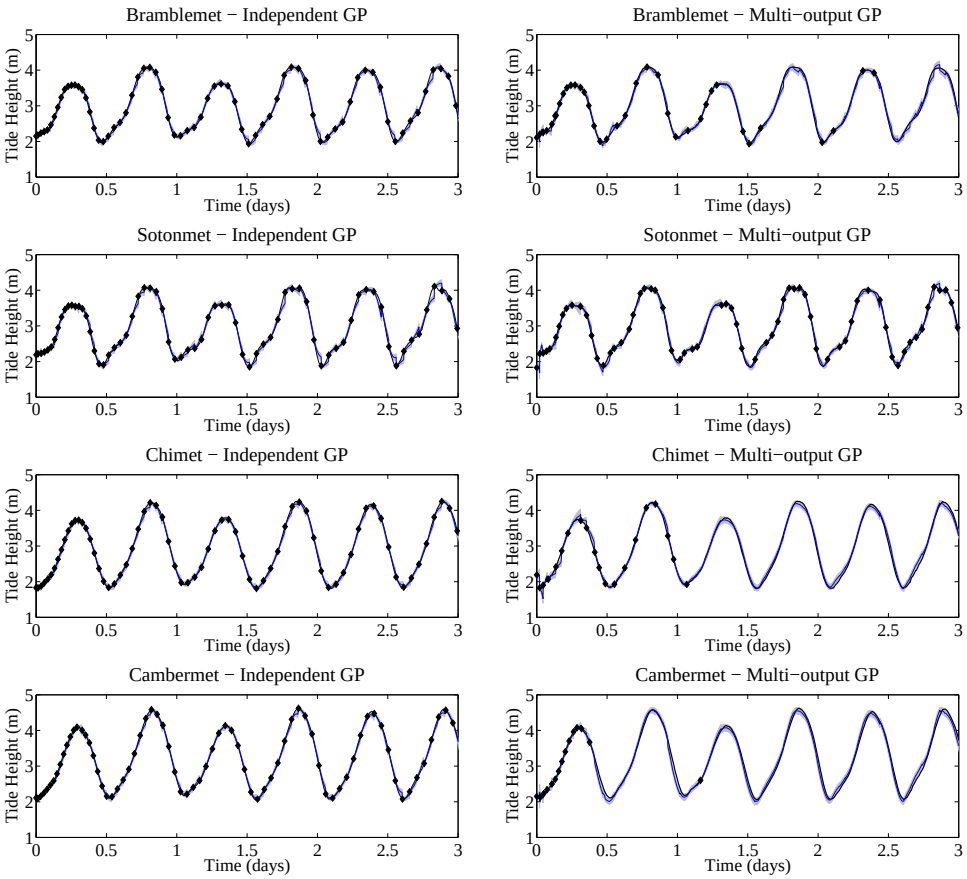

Fig. 8. Comparison of active sampling of tide data using (a) independent and (b) multi-output Gaussian processes.

time t, the GP must decide when next to observe, and from which sensor, given knowledge only of the observations recorded prior to t, in an attempt to maintain the uncertainty in tide height below 10cm.

Consider first the case shown in figure 8(a), in which separate independent GPs are used to represent each sensor. Note that a large number of observations are taken initially as the dynamics of the sensor readings are learnt, followed by a low but constant rate of observation. In contrast, for the multi-output case shown in figure 8(b), the GP is allowed to explicitly represent correlations and delays between the sensors. This data set is notable for the slight delay of the tide heights at the Chimet and Cambermet sensors relative to the Sotonmet and Bramblemet sensors, due to the nature of tidal flows in the area. Note that after an initial learning phase as the dynamics, correlations, and delays are inferred, the GP chooses to sample

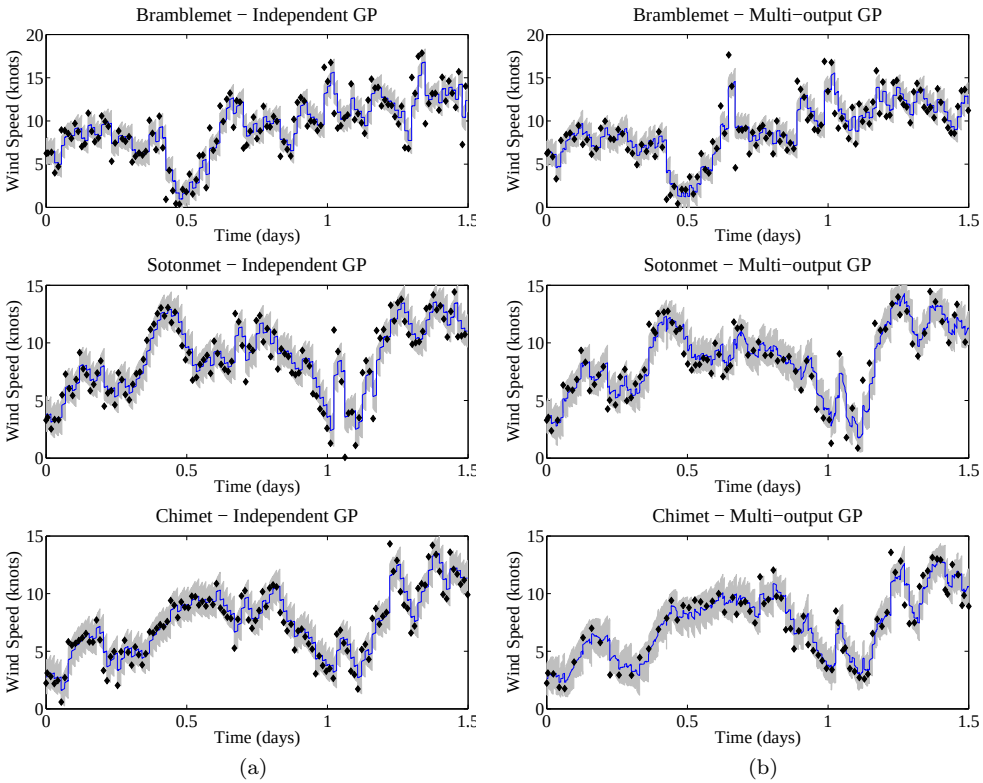

Fig. 9. Comparison of active sampling of wind speed using (a) independent and (b) multi-output Gaussian processes.

predominantly from the undelayed Sotonmet and Bramblemet sensors[3]. Despite no observations of the Chimet sensor being made within the time span plotted, the resulting predictions remain remarkably accurate. Consequently only 119 observations are required to keep the uncertainty below the specified tolerance, whereas 358 observations were required in the independent case. This represents another clear demonstration of how our prediction is able to benefit from the readings of multiple sensors.

Figure 9 shows similar results for the wind speed measurements from three of the four sensors (the Cambermet sensor being faulty during this period) where the goal was to maintain the uncertainty in wind speed below 1.5 knots. In this case, for purposes of clarity, the fine-grained data is not shown on the plot. Note that the measurement noise is much greater in this case, and this is reflected in the uncertainty in the GP predictions. Furthermore, note that while the Sotonmet and Chimet sensors exhibit a noticeable correlation, Bramblemet appears to be

[3]The dynamics of the tide height at the Sotonmet sensor are more complex than the other sensors due to the existence of a 'young flood stand' and a 'double high tide' in Southampton. For this reason, the GP selects Sotonmet as the most informative sensor and samples it most often.

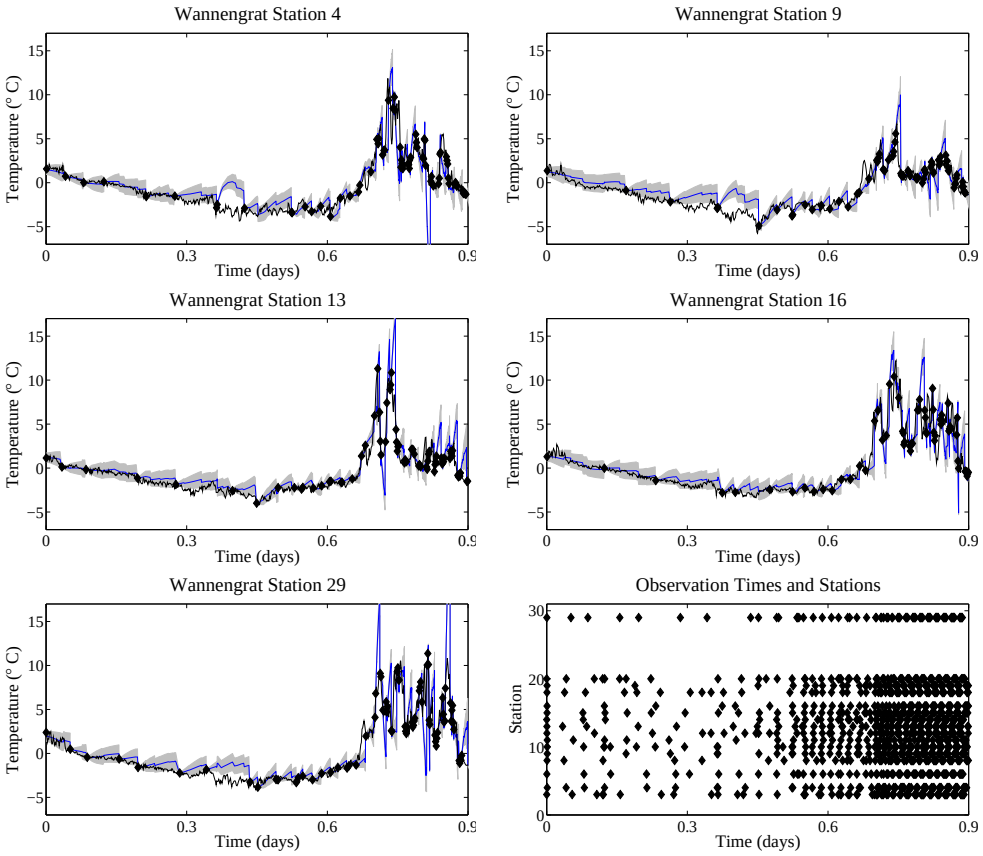

Fig. 10. Active sampling of ambient temperatures at 16 Wannengrat sensor stations.

relatively uncorrelated with both. This observation is reflected in the sampling that the GP performs. The independent GPs sample the Bramblemet, Sotonmet and Chimet sensors 126, 120 and 121 times respectively, while over the same period, our multi-output GP samples the same sensors 115, 88 and 81 times. Our multi-output GP learns on-line that the wind speed measurements of the Sotonmet and Chimet sensors are correlated, and then exploits this correlation in order to reduce the number of times that these sensors are sampled (inferring the wind speed at one location from observations of another). However, there is little or no correlation between the Bramblemet sensor and the other sensors, and thus, our multi-output GP samples Bramblemet almost as often as the independent GPs.

To investigate the practicality of our algorithm on large sensor networks, we apply it to data from 16 air temperature sensors from the Wannengrat Alpine Observatory. Given the larger number of sensors in this dataset, it is impractical to use the arbitrary parameterisation presented in equation (3), which requires a hyperparameter for every distinct pair of sensors. However, we can readily express the covariance between sensors as a function of their spatial separation, and thus,

		Data Points (N)		
		10	100	500
Hyperparameter Samples (η)	1	< 0.01	< 0.01	0.04
	10	0.02	0.02	0.20
	100	0.14	0.22	2.28
	1000	1.42	2.22	29.73

Table I. Required computation time (seconds) per update, over N the number of stored data points and η the number of hyperparameter samples. Experiments performed using MATLAB on a 3.00GHz processor with 2GB of RAM.

we use the Matérn covariance function given in equation (5) for this purpose, with the spatial distance being used to fill the role of r. This spatial distance was specified by a single isotropic scale hyperparameter, which was marginalised along with the other hyperparameters. Figure 10 shows the detailed predictions and samples for five of the sensors, and a summary plot showing the sample times of all 16 sensors. Note that the sensor sampling rate is relatively low at first, however, as the volatile fluctuations in temperature begin to occur at about $t = 0.7$ days, the sampling frequency automatically increases to ensure that the prediction uncertainty is maintained below the specified threshold.

6. COMPUTATION TIME

As described earlier, a key requirement of our algorithm is computational efficiency, in order that it can be used to represent multiple correlated sensors, and hence, used for real-time information processing. Here we consider the computation times involved in producing the results presented in the previous section. To this end, table I tabulates the computation times required in order to update the algorithm as a new observation is received. This computation time represents the cost of updating the weights of equation (17) and the Cholesky factor of $\mathbf{V}$ (as described in section 3.4). Once this calculation has been performed, making predictions at any point in time is extremely fast since it is simply a matter of adding another element in $\boldsymbol{x}_\star$.

Note that we expect the cost of computation to grow as $O(N^2)$ in the number of stored data points. Our proposed algorithm will automatically determine the quantity of data to store in order to achieve the desired level of accuracy. In the problems we have studied, a few hundred points were typically sufficient (the largest number we required was 750, for the multi-output wind speed data), although of course this will depend critically on the nature of the variables under consideration. Note also that the cost of computing equation (25) will grow in the cube of the number of samples in each hyperparameter. However, we consider only a fixed set of samples in each hyperparameter, and thus, equation (25) need only be computed once, off-line. In this case, our on-line costs are limited by the multiplication of that term by the likelihoods $\boldsymbol{r}_s$ to give the weights of equation (17), and this only grows as $O(\eta^2)$. Furthermore, note that this cost is independent of how the η samples are distributed amongst the hyperparameters.

The results in table I indicate that real-time information processing is clearly feasible for the problem sizes that we have considered. In general, limiting the

number of hyperparameter samples is of critical importance to achieving practical computation. As such, we should exploit any and all prior information that we possess about the system to limit the volume of hyperparameter space that our GP is required to explore online. For example, an informative prior expressing that the tidal period allow this hyperparameter to be fixed in value, hence reducing the space of hyperparameters values that must be explored. Similarly, an offline analysis of any available training data will return sharply peaked posteriors over our hyperparameters that will further restrict the required volume to be searched over on-line. For example, we represent the tidal period hyperparameter with only a single sample on-line, so certain does training data make us of its value. Finally, a simpler and less flexible covariance model, with fewer hyperparameters, could be chosen if computational limitations become particularly severe. The completely general spherical parameterisation requires a correlation hyperparameter for each pair of variables, and is clearly only feasible for moderate numbers of variables. However, as we showed with the Wannengrat active data selection results, it is equally possible to assume a covariance over variable labels which is a function of the spatial separation between the sensors reading them – sensors that are physically close are likely to be strongly correlated – in which case we would require only enough hyperparameters to define this measure of separation. While a more complicated model will return better predictions, a simple one or two hyperparameter covariance may supply accuracy sufficient for our needs.

7. RELATED WORK

Gaussian process regression has a long history of use within geophysics and geospatial statistics (where the process is known as kriging [Cressie 1991]), but have only recently been applied within sensor networks. For example, they have been used to represent spatial correlations between sensors so that the near-optimal sensor placement (in terms of mutual information) can be determined, for modelling wireless propagation between sensor nodes subject to censored observations of received signal strength [Ertin 2007], and in the form of multi-variate Gaussian distributions to represent correlations between different sensors and sensor types for energy efficient querying of a sensor network [Deshpande et al. 2004]. They have also been used to represent temporal correlations between the readings from a single sensor in order to perform adaptive sampling — maximising the information gain subject to a sampling constraint [Kho et al. 2009].

Our work differs in that we use GPs to represent temporal correlations, and represent correlations and delays between sensors with additional hyperparameters. It is thus closely related to other work using GPs to perform regression over multiple responses [Boyle and Frean 2005; Teh et al. 2005]. However, our focus is to derive a computationally efficient algorithm, and thus, we use a number of novel computational techniques to allow the re-use of previous calculations as new sensor observations are made. We additionally use a novel Bayesian Monte Carlo technique to marginalise the hyperparameters that describe the correlations and delays between sensors. Finally, we use the variance of the GP's predictions in order to perform active data selection. Likewise, our work differs from that previously using censored sensor readings within a GP framework [Ertin 2007], since our work pro-

poses a principled Bayesian Monte Carlo method for adapting our models to the data, where Ertin's model assumes that the hyperparameters are known a priori, and hence, does not consider how likelihoods should be computed in this context. Furthermore, in our work, Monte Carlo techniques are also used to evaluate our other integrals, rather than taking Laplace approximations, allowing more accurate representation of the correlation amongst censored readings.

Finally, our approach has several advantages relative to sequential state-space models such as the Kalman filter [Girard et al. 2003; Jazwinski 1970] which have also been applied in environmental settings for tracking sensor readings [Bertino et al. 2003]. Firstly, these state-space models require the discretisation of the time input, representing a discarding of potentially valuable information. Secondly, their sequential nature means they must necessarily perform difficult iterations in order to manage missing or late data, or to produce long-range forecasts. In our GP approach, what observations we have are readily managed, regardless of when they were made. Equally, the computation cost of all our predictions is identical, irrespective of the time or place we wish to make them about. Finally, a sequential framework requires an explicit specification of a transition model. In our approach, we are able to learn a model from data even if our prior knowledge is negligible, and the benefits of our approach are empirically supported by the results presented in section 5.

8. CONCLUSIONS

In this paper we addressed the need for algorithms capable of performing real-time information processing of sensor network data, and we presented a novel computationally efficient formalism of a multi-output Gaussian process. Using weather data collected from two sensor networks, we demonstrated that this formalism allows an end-user to make effective use of sensor data even in the presence of network outages and sensor failures (recovering missing data by making predictions based on previous sensor readings and the current readings of sensors that are functioning), and to automatically determine the sampling rate of each individual sensor in order to ensure that the uncertainty in the environmental parameter being measured stays within a pre-specified limit.

Our future work in this area consists of three areas. First, as a potential replacement to the fixed hyperparameter samples used in this work, we would like to investigate the use of a moving set of hyperparameter samples. In such a scheme, both the weights and positions of samples would be adjusted according to data received, and as the posterior distributions of these hyperparameters become more sharply peaked, we would reduce the number of samples to further increase the computational efficiency of our algorithm.

Second, we intend to investigate the use of correlations between different sensor types (rather than between different sensors of the same type as presented here) to perform regression and prediction within our weather sensor network. Our formalism is in no way restricted to identical sensors, and it is expected that many sensors will exhibit correlations in this setting. For example, both wind speed and wind direction, and air temperature and air pressure are likely to be exhibit correlations as weather fronts move over the sensor network.

Finally, we would like to use the probabilistic model that the GP builds to automatically handle faulty and unreliable sensors within the network. Note that we do not wish to simply detect these faulty sensors., but would like to simultaneously perform both detection and prediction, despite the presence of these failures. Our preliminary work is this area indicates that nonstationary covariance functions that model such changes can be introduced to the formalism described here to achieve this, and have already been shown to work effectively on real-world sensor data derived sensor networks described in this paper [Garnett et al. 2009].

Acknowledgments

This research was undertaken as part of the ALADDIN (Autonomous Learning Agents for Decentralised Data and Information Networks) project and is jointly funded by a BAE Systems and EPSRC strategic partnership (EP/C548051/1). We would like to thank B. Blaydes of the Bramblemet/Chimet Support Group, and W. Heaps of Associated British Ports (ABP) for allowing us access to the weather sensor network, hosting our RDF data on the sensor web sites, and for providing raw sensor data as required.

REFERENCES

Abrahamsen, P. 1997. A review of Gaussian random fields and correlation functions. Tech. Rep. 917, Norwegian Computing Center, Box 114, Blindern, N-0314 Oslo, Norway. 2nd edition.

Barnaghi, P., Meissner, S., Presser, P., and Moessner, K. 2009. Sense and sens'ability: Semantic data modelling for sensor networks. In *Proceedings of the ICT Mobile and Wireless Communications Summit*. Santander, Spain.

Bertino, L., Evensen, G., and Vackernagel, H. 2003. Sequential data assimilation techniques in oceanography. *International Statistical Review* 71, 223–242.

Boyle, P. and Frean, M. 2005. Dependent Gaussian processes. In *Advances in Neural Information Processing Systems 17*. The MIT Press, 217–224.

Cressie, N. A. C. 1991. *Statistics for spatial data*. John Wiley & Sons.

Deshpande, A., Guestrin, C., Madden, S., Hellerstein, J., and Hong, W. 2004. Model-driven data acquisition in sensor networks. In *Proceedings of the Thirtieth International Conference on Very Large Databases*. Toronto, Canada, 588–599.

Ertin, E. 2007. Gaussian process models for censored sensor readings. In *Proceedings of the Fourteenth IEEE/SP Workshop on Statistical Signal Processing*. Madison, Wisconsin, USA, 665–669.

Fuentes, M., Chaudhuri, A., and Holland, D. H. 2007. Bayesian entropy for spatial sampling design of environmental data. *Environmental and Ecological Statistics* 14, 323–340.

Garnett, R., Osborne, M. A., and Roberts, S. J. 2009. Sequential Bayesian prediction in the presence of changepoints. In *Proceedings of the Twenty-Sixth International Conference on Machine Learning*. Montreal, Canada.

Genz, A. 1992. Numerical computation of multivariate normal probabilities. *Journal of Computational and Graphical Statistics 1,* 2, 141–149.

Girard, A., Rasmussen, C., Candela, J., and Murray-Smith, R. 2003. Gaussian process priors with uncertain inputs – application to multiple-step ahead time series forecasting. In *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 545–552.

Hart, J. K. and Martinez, K. 2006. Environmental sensor networks: A revolution in the earth system science? *Earth-Science Reviews 78*, 177–191.

Jazwinski, A. 1970. *Stochastic processes and filtering theory*. Academic Press New York.

Kho, J., Rogers, A., and Jennings, N. R. 2009. Decentralized control of adaptive sampling in wireless sensor networks. *ACM Transactions on Sensor Networks 5,* 3, 1–35.

Krause, A., Guestrin, C., Gupta, A., and Kleinberg, J. 2006. Near-optimal sensor placements: Maximizing information while minimizing communication cost. In *Proceedings of the Fifth International Conference on Information Processing in Sensor Networks*. Nashville, Tennessee, USA, 2–10.

Lassila, O. and Swick, R. R. 1999. Resource description framework (RDF) model and syntax specification. Available at `http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/`.

Lee, S. M. and Roberts, S. J. 2008. Multivariate time series forecasting in incomplete environments. Tech. Rep. PARG-08-03. Available at `www.robots.ox.ac.uk/~parg/publications.html`, University of Oxford.

MacKay, D. J. C. 2002. *Information Theory, Inference & Learning Algorithms*. Cambridge University Press.

O'Hagan, A. 1987. Monte Carlo is fundamentally unsound. *The Statistician 36*, 247–249.

Padhy, P., Dash, R. K., Martinez, K., and Jennings, N. R. 2010. A utility-based adaptive sensing and multi-hop communication protocol for wireless sensor networks. *ACM Transactions on Sensor Networks 6,* 3. In print.

Pinheiro, J. and Bates, D. 1996. Unconstrained parameterizations for variance-covariance matrices. *Statistics and Computing 6*, 289–296.

Rasmussen, C. E. and Ghahramani, Z. 2003. Bayesian Monte Carlo. In *Advances in Neural Information Processing Systems 15*. The MIT Press, 489–496.

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

Sasena, M. J. 2002. Flexibility and Efficiency Enhancements for Constrained Global Design Optimization with Kriging Approximations. Ph.D. thesis, University of Michigan.

Stein, M. 2005. Space-Time Covariance Functions. *Journal of the American Statistical Association 100,* 469, 310–322.

Teh, Y. W., Seeger, M., and Jordan, M. I. 2005. Semiparametric latent factor models. In *Proceedings of the Conference on Artificial Intelligence and Statistics*. Barbados, 333–340.

The MathWorks. 2007. MATLAB R2007a. Natick, MA.

A. APPENDIX

A.1 Cholesky Factor Update

We have a positive definite matrix, represented in block form as $\begin{bmatrix} V_{1,1} & V_{1,3} \\ V_{1,3}^{\mathrm{T}} & V_{3,3} \end{bmatrix}$ and its Cholesky factor, $\begin{bmatrix} R_{1,1} & R_{1,3} \\ 0 & R_{3,3} \end{bmatrix}$. Given a new positive definite matrix, which differs from the old only in the insertion of some new rows and columns, $\begin{bmatrix} V_{1,1} & V_{1,2} & V_{1,3} \\ V_{1,2}^{\mathrm{T}} & V_{2,2} & V_{2,3} \\ V_{1,3}^{\mathrm{T}} & V_{2,3}^{\mathrm{T}} & V_{3,3} \end{bmatrix}$, we wish to efficiently determine its Cholesky factor, $\begin{bmatrix} S_{1,1} & S_{1,2} & S_{1,3} \\ 0 & S_{2,2} & S_{2,3} \\ 0 & 0 & S_{3,3} \end{bmatrix}$. For $\mathbf{A}$ triangular, we define $\boldsymbol{x} = \mathbf{A} \setminus \boldsymbol{b}$ as the solution to the equations $\mathbf{A}\,\boldsymbol{x} = \boldsymbol{b}$ as found by the use of backwards or

forwards substitution. The following rules are readily obtained

$$S_{1,1} = R_{1,1} \tag{26}$$
$$S_{1,2} = R_{1,1}^{\mathrm{T}} \setminus V_{1,2} \tag{27}$$
$$S_{1,3} = R_{1,3} \tag{28}$$
$$S_{2,2} = \mathrm{chol}(V_{2,2} - S_{1,2}^{\mathrm{T}} S_{1,2}) \tag{29}$$
$$S_{2,3} = S_{2,2}^{\mathrm{T}} \setminus (V_{2,3} - S_{1,2}^{\mathrm{T}} S_{1,3}) \tag{30}$$
$$S_{3,3} = \mathrm{chol}(R_{3,3}^{\mathrm{T}} R_{3,3} - S_{2,3}^{\mathrm{T}} S_{2,3})\,. \tag{31}$$

By setting the appropriate row and column dimensions (to zero if necessary), this allows us to efficiently determine the Cholesky factor given the insertion of rows and columns in any position.

A.2 Data Term Update

We have all terms defined in Section A.1, in addition to $\begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \end{bmatrix}$ and the product $\begin{bmatrix} C_1 \\ C_3 \end{bmatrix} \triangleq \begin{bmatrix} R_{1,1} & R_{1,3} \\ 0 & R_{3,3} \end{bmatrix}^{-1} \begin{bmatrix} Y_1 \\ Y_3 \end{bmatrix}$. To efficiently determine $\begin{bmatrix} D_1 \\ D_2 \\ D_3 \end{bmatrix} \triangleq \begin{bmatrix} S_{1,1} & S_{1,2} & S_{1,3} \\ 0 & S_{2,2} & S_{2,3} \\ 0 & 0 & S_{3,3} \end{bmatrix}^{-1} \begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \end{bmatrix}$, we have

$$D_1 = C_1 \tag{32}$$
$$D_2 = S_{2,2}^{-\mathrm{T}} \left(Y_2 - S_{1,2}^{\mathrm{T}}\, C_1\right) \tag{33}$$
$$D_3 = S_{3,3}^{-\mathrm{T}} \left(R_{3,3}^{\mathrm{T}}\, C_3 - S_{2,3}^{\mathrm{T}}\, D_2\right). \tag{34}$$

A.3 Cholesky Factor Downdate

We have a positive definite matrix, represented in block form as $\begin{bmatrix} V_{1,1} & V_{1,2} & V_{1,3} \\ V_{1,2}^{\mathrm{T}} & V_{2,2} & V_{2,3} \\ V_{1,3}^{\mathrm{T}} & V_{2,3}^{\mathrm{T}} & V_{3,3} \end{bmatrix}$ and its Cholesky factor, $\begin{bmatrix} S_{1,1} & S_{1,2} & S_{1,3} \\ 0 & S_{2,2} & S_{2,3} \\ 0 & 0 & S_{3,3} \end{bmatrix}$. Given a new positive definite matrix, which differs from the old only in the deletion of some new rows and columns, $\begin{bmatrix} V_{1,1} & V_{1,3} \\ V_{1,3}^{\mathrm{T}} & V_{3,3} \end{bmatrix}$, we wish to efficiently determine its Cholesky factor $\begin{bmatrix} R_{1,1} & R_{1,3} \\ 0 & R_{3,3} \end{bmatrix}$. The following rules are readily obtained

$$R_{1,1} = S_{1,1} \tag{35}$$
$$R_{1,3} = S_{1,3} \tag{36}$$
$$R_{3,3} = \mathrm{chol}(S_{2,3}^{\mathrm{T}} S_{2,3} + S_{3,3}^{\mathrm{T}} S_{3,3})\,. \tag{37}$$

Note that the special structure of equation (37) can be exploited for the efficient resolution of the required Cholesky operation, as, for example, in the MATLAB function cholupdate [The MathWorks 2007]. By setting the appropriate row and column dimensions (to zero if necessary), this allows us to efficiently determine the Cholesky factor given the deletion of rows and columns in any position.