Towards Real-Time Information Processing of Sensor Network Data using Computationally Efficient Multi-output Gaussian Processes

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

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

Abstract

In this paper, we describe a novel, computationally efficient algorithm that facilitates the autonomous acquisition of readings from sensor networks (deciding when and which sensor to acquire readings from at any time), and which can, with minimal domain knowledge, perform a range of information processing tasks including modelling the accuracy of the sensor readings, predicting the value of missing sensor readings, and predicting how the monitored environmental variables will evolve into the future. Our motivating scenario is the need to provide situational awareness support to first responders at the scene of a large scale incident, and to this end, we describe a novel iterative formulation of a multi-output Gaussian process that can build and exploit a probabilistic model of the environmental variables being measured (including the correlations and delays that exist between them). We validate our approach using data collected from a network of weather sensors located on the south coast of England.

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 typically make current data available through the internet, and thus, are increasingly being used for the real-time monitoring of environmental events such as floods or storm events (see [7] for a review of such environmental sensor networks).

Such real-time access to sensor data is also a feature of *pervasive sensor systems* in which sensors owned by multiple stakeholders (e.g. private individuals, building owners, and local authorities) are ubiquitously deployed within urban environments and make their information available to multiple users directly through standard web interfaces (see the CitySense project of Harvard University [13] and Microsoft's SenseWeb project [1]). Such networks have many applications, including traffic or pollution monitoring, and within the ALADDIN project (`http://www.aladdinproject.org`), we are seeking to use such networks to provide situational awareness support to first responders at the scene of a large scale incident. We envisage providing these first responders with a mobile computer or personal digital assistant (PDA) that is capable of collecting information from local sensors, compiling a coherent world view, and then assisting in decision making. An example application would be to provide fire fighters with local weather information, and to predict future wind changes through observations of nearby sensors. Other applications include tracking the movement of dangerous gas, chemical or smoke plumes, and monitoring the structural integrity of buildings after an earthquake.

Using real-time sensor data in this manner presents many novel challenges; not least the need for self-describing data formats, and standard protocols such that sensors can advertise their existence and capabilities to potential users. However, more significantly for us, many of the information processing tasks that would previously have been performed off-line by the owner or single user of an environmental sensor network (such as detecting faulty sensors, fusing noisy measurements from several sensors, and deciding how frequently readings should be taken), must now be performed in real-time on the mobile computers and PDAs carried by the multiple different users of the system (who may have different goals and may be using sensor readings for very different tasks). Furthermore, to support decision making, it may also be necessary to use the trends and correlations observed in previous data to predict the value of environmental parameters into the future, or to predict the reading

of a sensor that is temporarily unavailable (e.g. due to network outages). Finally, we note that the open nature of the network (in which additional sensors may be deployed, and existing sensors may be removed, repositioned or updated at any time) means that these 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 [4] and Gaussian processes [18]. 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 [6], 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 [10]). Thus, there is a clear need for more computationally efficient algorithms, that can be deployed on the mobile computers and PDAs carried by our first responders, in order to perform this information processing 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 Gaussian process (GP) that uses a computationally efficient implementation of *Bayesian Monte Carlo* to marginalise hyperparameters, efficiently re-uses previous 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 (including the correlations and delays that exist between them). 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 a network of weather sensors on the south coast of England, 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 sensor network used to validate this formulation. In section 5 we present experimental results using data from this network, 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 mobile device or PDA on which the algorithm is running, or alternatively, where the algorithm is actually controlling 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, $\{[[l_1, t_1], y_1], \ldots, [[l_N, t_N], y_N]\}$, in which we, for example, observe the value y_1 for the $l_1{}^{\text{th}}$ variable at time t_1, whose true unknown value is x_1. 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 $\boldsymbol{y}_D \triangleq [y_1, \ldots, y_N]$ of variables labelled by $\boldsymbol{l}_D \triangleq [l_1, \ldots, l_N]$ at times $\boldsymbol{t}_D \triangleq [t_1, \ldots, t_N]$. Given this data, we are interested in inferring the vector of values $\boldsymbol{x}_\star$ 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 have often been addressed using multi-layer neural networks. However, Gaussian processes (GPs) are increasingly being applied in this area. They represent a powerful way to perform Bayesian inference about functions; we consider our environmental variables as just such a function [18]. This function takes as inputs the variable label and time pair $[l, t]$ and produces as output the variable's value x. 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 I. A GP is then a generalised multivariate Gaussian prior distribution over the (potentially infinite number of) outputs of this function:

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

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 [18] for example), there is much scope for the development of computationally efficient implementations. 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. Space precludes a full description of this algorithm (see [15] for the full details), however, 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 [2, 20], 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:

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

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

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

where $\boldsymbol{g}$ gives an intuitive length scale for each environmental variable, and $\mathbf{s}^T\mathbf{s}$ is the correlation matrix [16]. This allows us to represent any possible degree of correlation between our variables.

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 where we expect our variable to be well-represented by the superposition of an oscillatory and a non-oscillatory component. We represent both terms using the Matérn class [18] (with $\nu = 5/2$), given by:

$$K_{\text{time}}(t, t') \triangleq h^2 \left(1 + \sqrt{5}r + \frac{5r^2}{3}\right) \exp\left(-\sqrt{5}r\right) \quad (4)$$

where $r = \left|\frac{t-t'}{w}\right|$ for non-periodic terms, and $r = \sin\pi\left|\frac{t-t'}{w}\right|$ for periodic ones. 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:

$$V([l, t], [l', t']) \triangleq K([l, t], [l', t']) + \sigma^2\, \delta\big([l, t] - [l', t']\big) \quad (5)$$

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.

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:

$$p(\boldsymbol{x}_\star | \, \boldsymbol{y}_D, \, \phi, \, I) = \mathbf{N}(\boldsymbol{x}_\star; \boldsymbol{m}_\star, \mathbf{C}_\star) \tag{6}$$

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

$$\begin{aligned} \boldsymbol{m}_\star &= \boldsymbol{\mu}_\phi(\boldsymbol{z}_\star) + \mathbf{K}_\phi(\boldsymbol{z}_\star, \boldsymbol{z}_D)\mathbf{V}_\phi(\boldsymbol{z}_D, \boldsymbol{z}_D)^{-1}(\boldsymbol{y}_D - \boldsymbol{\mu}_\phi(\boldsymbol{z}_D)) \\ \mathbf{C}_\star &= \mathbf{K}_\phi(\boldsymbol{z}_\star, \boldsymbol{z}_\star) - \mathbf{K}_\phi(\boldsymbol{z}_\star, \boldsymbol{z}_D)\mathbf{V}_\phi(\boldsymbol{z}_D, \boldsymbol{z}_D)^{-1}\mathbf{K}_\phi(\boldsymbol{z}_D, \boldsymbol{z}_\star) \end{aligned} \tag{7}$$

3.2 Marginalisation

Of course, it is rare that we can be certain a priori about the values of our hyperparameters. Rather than equation (6), the quantity of our interest is actually:

$$p(\boldsymbol{x}_\star | \, \boldsymbol{y}_D, I) = \frac{\int p(\boldsymbol{x}_\star | \, \boldsymbol{y}_D, \phi, I) \, p(\boldsymbol{y}_D | \, \phi, I) \, p(\phi | \, I) \, \mathrm{d}\phi}{\int p(\boldsymbol{y}_D | \, \phi, I) \, p(\phi | \, I) \, \mathrm{d}\phi} \tag{8}$$

in which we have marginalised ϕ. Unfortunately, both our likelihood $p(\boldsymbol{y}_D | \, \phi, I)$ and predictions $p(\boldsymbol{x}_\star | \, \boldsymbol{y}_D, \phi, I)$ 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:

$$\begin{aligned} q(\phi) &\triangleq p(\boldsymbol{x}_\star | \, \boldsymbol{y}_D, \, \phi, \, I) \\ r(\phi) &\triangleq p(\boldsymbol{y}_D | \, \phi, \, I) \end{aligned} \tag{9}$$

at a set of sample points $\boldsymbol{\phi}_S = [\phi_i, \ldots, \phi_\eta]$, 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. Clearly, we can't afford to evaluate the functions q and r for all possible ϕ. However, we can view this sparse sampling as 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 a second GP prior to these functions [17]. We can then use our computed samples $\boldsymbol{q}_S$ in order to perform regression about the value of $q(\phi_\star)$ for any other $\phi_\star$ of interest, and similarly for r. 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:

$$p(\phi | \, I) \triangleq \mathbf{N}\big(\phi; \boldsymbol{\nu}, \lambda^{\mathrm{T}}\lambda\big) \tag{10}$$

We then assign a *squared exponential* covariance function for the GP over both q and r given by:

$$K(\phi, \phi') \triangleq \mathbf{N}\big(\phi; \phi', \mathbf{w}^{\mathrm{T}}\mathbf{w}\big) \tag{11}$$

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

$$\mathfrak{N}_S(i,j) \triangleq \mathbf{N}\left(\begin{bmatrix} \boldsymbol{\phi}_i \\ \boldsymbol{\phi}_j \end{bmatrix}; \begin{bmatrix} \boldsymbol{\nu} \\ \boldsymbol{\nu} \end{bmatrix}, \begin{bmatrix} \lambda^{\mathrm{T}}\lambda + \mathbf{w}^{\mathrm{T}}\mathbf{w} & \lambda^{\mathrm{T}}\lambda \\ \lambda^{\mathrm{T}}\lambda & \lambda^{\mathrm{T}}\lambda + \mathbf{w}^{\mathrm{T}}\mathbf{w} \end{bmatrix} \right) \tag{12}$$

it can be shown that:

$$p(\boldsymbol{x}_\star | \, \boldsymbol{y}_D, I) \simeq \boldsymbol{q}_S^{\mathrm{T}} \frac{\mathbf{K}(\boldsymbol{\phi}_S, \boldsymbol{\phi}_S)^{-1} \, \mathfrak{N}_S \, \mathbf{K}(\boldsymbol{\phi}_S, \boldsymbol{\phi}_S)^{-1} \, \boldsymbol{r}_S}{\mathbf{1}_{S,1}^{\mathrm{T}} \, \mathbf{K}(\boldsymbol{\phi}_S, \boldsymbol{\phi}_S)^{-1} \, \mathfrak{N}_S \, \mathbf{K}(\boldsymbol{\phi}_S, \boldsymbol{\phi}_S)^{-1} \, \boldsymbol{r}_S} \tag{13}$$

where $\mathbf{1}_{S,1}$ is a vector containing only ones of dimensions equal to $\boldsymbol{q}_S$. Note that equation (13) can be viewed as a linear combination of the elements of $\boldsymbol{q}_S$. With a GP on $p(\boldsymbol{x}_\star | \, \phi, \, I)$, each $q_i = p(\boldsymbol{x}_\star | \, \boldsymbol{y}_D, \, \phi_i, \, I)$ will be a slightly different Gaussian. Hence we effectively approximate $p(\boldsymbol{x}_\star | \, \boldsymbol{y}_D, I)$ as a Gaussian (process) mixture; Bayesian Monte Carlo returns a weighted sum of our predictions evaluated at a sample set of hyperparameters. Unlike traditional Monte Carlo, these weights are informed not only by our function values $\boldsymbol{q}_S$ and $\boldsymbol{r}_S$, but also by their arguments $\boldsymbol{\phi}_S$. Bayesian Monte Carlo makes the best use of all pertinent information [14]. As such, it avoids the risk of overfitting that occurs when applying a less principled technique such as likelihood maximisation [12].

3.3 Efficient Implementation

The most stable implementation of equation (7) involves the use of the Cholesky decomposition, $\mathbf{R}(\boldsymbol{z}_D, \boldsymbol{z}_D)$, of $\mathbf{V}(\boldsymbol{z}_D, \boldsymbol{z}_D)$, such that $\mathbf{V}(\boldsymbol{z}_D, \boldsymbol{z}_D) = \mathbf{R}(\boldsymbol{z}_D, \boldsymbol{z}_D)^{\mathrm{T}} \, \mathbf{R}(\boldsymbol{z}_D, \boldsymbol{z}_D)$. 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). 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 (13). 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.2 for the details of this operation). This allows us to rapidly remove unwanted data, compute our uncertainty about $\boldsymbol{x}_\star$, 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 a number 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 (13). 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 (13), 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. [19]):

$$K(\phi,\, \phi') = \prod_e K_e(\phi_{(e)},\, \phi'_{(e)}) \tag{14}$$

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 (13) reduces to the Kronecker product of the equivalent term over each individual hyperparameter:

$$\begin{aligned}
&\mathbf{K}(\boldsymbol{\phi}_S, \boldsymbol{\phi}_S)^{-1}\, \mathfrak{N}_S\, \mathbf{K}(\boldsymbol{\phi}_S, \boldsymbol{\phi}_S)^{-1} = \\
&\quad \mathbf{K}(\boldsymbol{\phi}_{(1),S}, \boldsymbol{\phi}_{(1),S})^{-1} \mathfrak{N}_S(\boldsymbol{\phi}_{(1),S}, \boldsymbol{\phi}_{(1),S}) \mathbf{K}(\boldsymbol{\phi}_{(1),S}, \boldsymbol{\phi}_{(1),S})^{-1} \\
&\otimes \mathbf{K}(\boldsymbol{\phi}_{(2),S}, \boldsymbol{\phi}_{(2),S})^{-1} \mathfrak{N}_S(\boldsymbol{\phi}_{(2),S}, \boldsymbol{\phi}_{(2),S}) \mathbf{K}(\boldsymbol{\phi}_{(2),S}, \boldsymbol{\phi}_{(2),S})^{-1} \\
&\otimes \dots
\end{aligned} \tag{15}$$

This means that we only have to perform the Cholesky factorisation and multiplication with matrices whose size equals the number of samples for each hyperparameter. For example, if we use, say, 100 samples for each of our 20 hyperparameters, we only ever need to perform our expensive $O(\eta^3)$ operations on matrices of size 100, rather than on the full matrix of size 100^{20}. Thus, this 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.

3.4 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 prespecified 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.

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 this is possible due to the fact that the uncertainty of a single GP, as given by $\mathbf{C}_\star$ in equation (7), is actually dependent only on the location of a 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

```
<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>
```

Figure 1: Example RDF data from the Bramblemet sensor.

data we might possibly collect. Hence 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 a network of weather sensors located on the south coast of England[1]. This 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 `http://www.bramblemet.co.uk`). The use of such weather sensors is attractive since they have immediate application within our motivating disaster response scenario, they exhibit challenging correlations and delays whose physical processes are well understood, and they are subject to network outages that generate instances of missing sensor readings on which we can evaluate our information processing algorithms.

To facilitate the autonomous collection of sensor data by our information processing algorithm, we have supplemented each sensor web page with machine readable RDF data (see figure 1 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

[1]The network is maintained by the Bramblemet/Chimet Support Group and funded by organisations including the Royal National Lifeboat Institution, Solent Cruising and Racing Association and Associated British Ports.

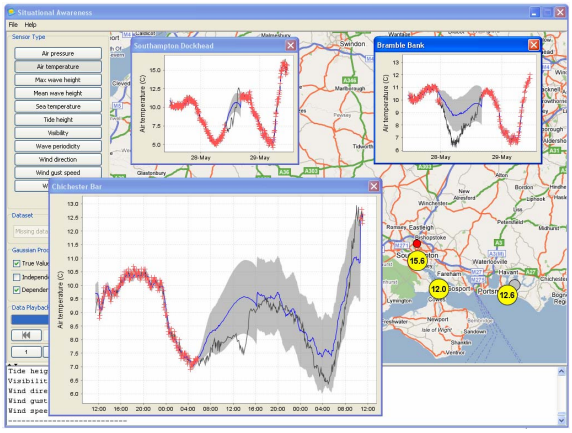

Figure 2: Java implementation of our information processing algorithm.

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 [11, 21]. 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 the Microsoft SenseWeb project [1], for an example ontology that defines a hierarchy of sensor types).

Finally, in order to visualise the sensor data and the predictions of our information processing algorithm, we have implemented a Java prototype of the software that will run on the mobile computer or PDA carried by our first responders to provide situational awareness support (see figure 2).

5 Empirical Evaluation

In this section we empirically evaluate our information processing algorithm on real weather data collected from the sensor network described above. We compare our multi-output GP formalism against conventional independent GPs in which each environmental variable is modelled separately (i.e. correlations between these parameters are ignored). In this comparison, we present results for three different sensor types: tide height, air temperature and wind speed. 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

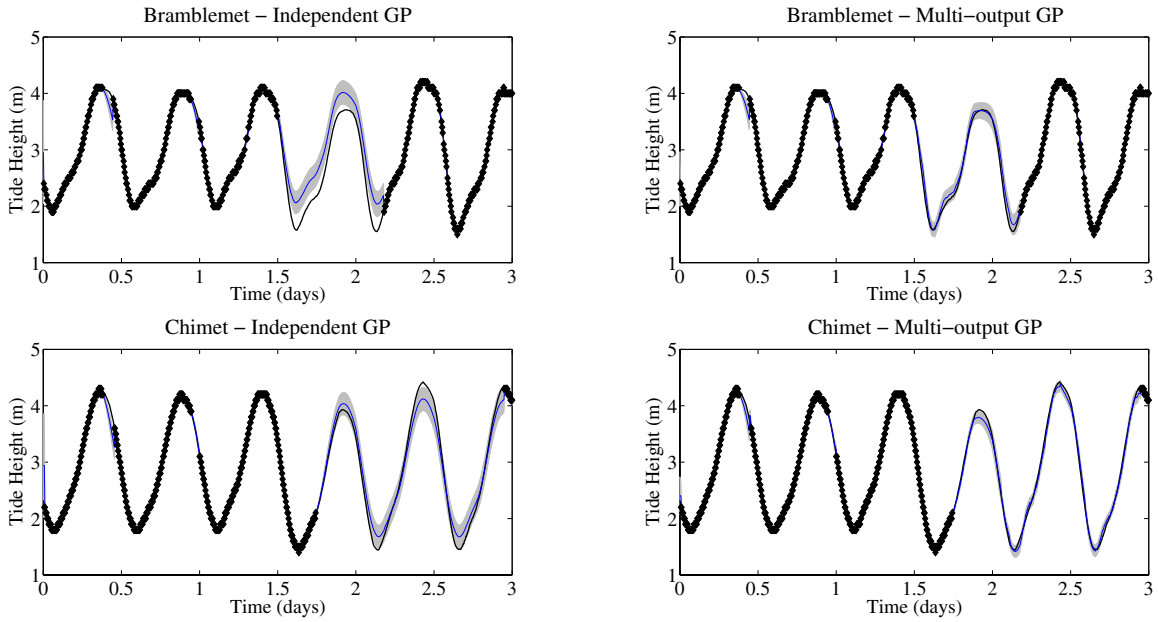

Figure 3: Prediction and regression of tide height data for (a) independent and (b) multi-output Gaussian processes.

sensor readings. Air temperature and wind speed were chosen since they exhibit a very different noise and correlation structure to the tide height measurements, and thus demonstrate that the generic approach describe here is still able to perform reliable regression and prediction.

5.1 Regression and Prediction

Figures 3 and 4 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, these figures depict the posterior distribution of the GP, conditioned on all observations prior to t.

We first consider figure 3 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 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 4 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.

5.2 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 5, and for three wind speed sensors in figure 6. Again, these plots depict dynamic choices; at time t, the GP must decide when next to observe, and from which sensor, given knowledge only of the observations recorded

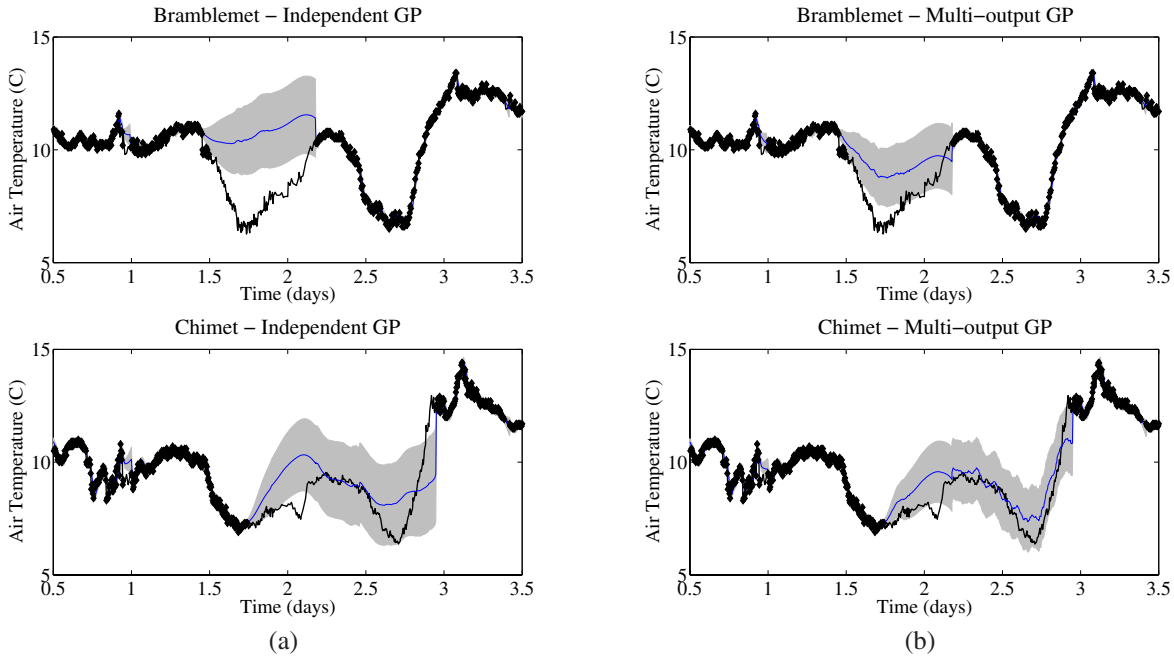

Figure 4: Prediction and regression of air temperature data for (a) independent and (b) multi-output Gaussian processes.

prior to t, in an attempt to maintain the uncertainty in tide height below 10cm.

Consider first the case shown in figure 5(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 5(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 predominantly from the undelayed Sotonmet and Bramblemet sensors[2]. 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 6 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 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.

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-

[2]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.

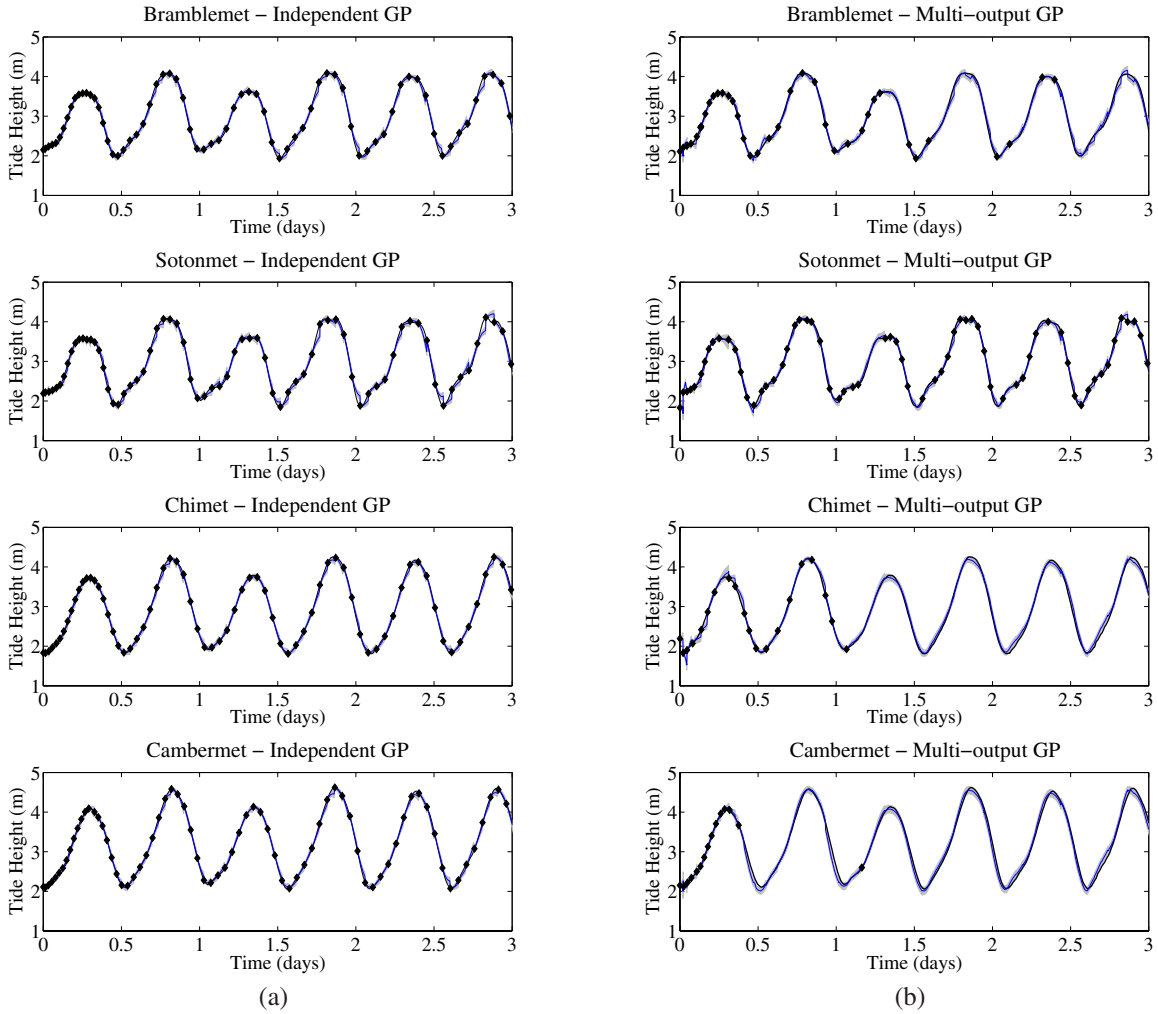

Figure 5: Comparison of active sampling of tide data using (a) independent and (b) multi-output Gaussian processes.

time information processing. Here we consider the computation times involved in producing the results presented in the previous section. To this end, table 1 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 (13) and the Cholesky factor of $\mathbf{V}$ (as described in section 3.3). Once this calculation has been performed, making predictions at any point in time is extremely fast (it is simply a matter of adding another element in $\boldsymbol{z}_\star$).

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 (15) 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 (15) 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 (13), 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 1 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

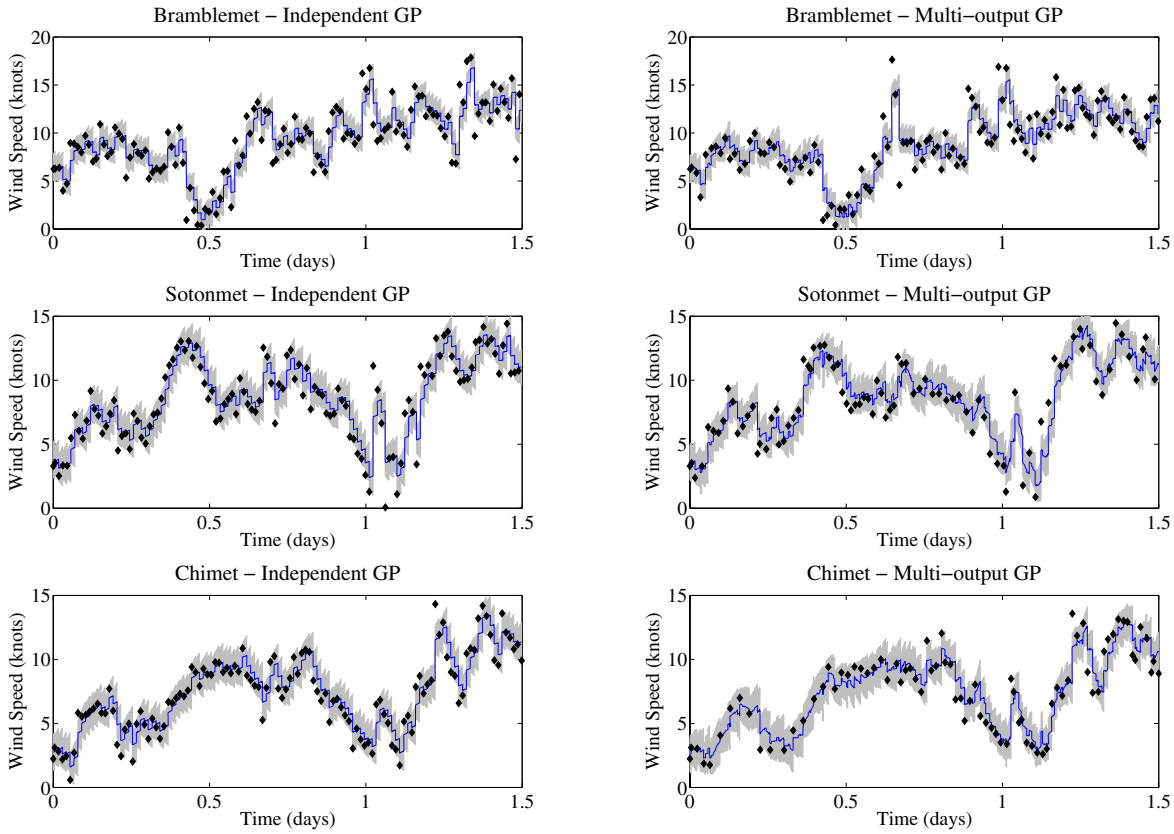

Figure 6: Comparison of active sampling of wind speed using (a) independent and (b) multi-output Gaussian processes.

expressing that the tidal period is likely to be around half a day will greatly reduce the number of samples required for this hyperparameter. 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. Note that the use of the completely general spherical parameterisation requires a correlation hyperparameter for each pair of variables, an approach which is clearly only feasible for moderate numbers of variables. A simple alternative, of course, would be to assume a covariance over variable label 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 [4]), but has only recently been applied within sensor networks. Examples here include the use of GPs to represent spatial correlations between sensors in order that they may be positioned to maximise mutual information [10], and the use of multi-variate Gaussians to represent correlations between different sensors and sensor types for energy efficient querying of a sensor network [5].

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 [3, 22]. 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. Furthermore, we use Bayesian Monte Carlo techniques to marginalise the hyperparameters that describe the correlations and delays between sensors, and finally, we use the variance of the GP's predictions in order to perform

		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 1: 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.

active data selection.

Our approach has several advantages relative to sequential state-space models [8, 9], Firstly, these 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.

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 a sensor network on the south coast of the UK, we demonstrated that this formalism could effectively predict missing sensor readings caused by network outages, and could perform active sampling to maintain estimation uncertainty below a pre-specified threshold.

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. In addition, we would like to use our probabilistic model to automatically detect failed or unreliable sensors within the network.

Figure 7: Prototype deployment of an information processing algorithm on a PDA, and a stand-alone weather sensor with which it can directly communicate through Wi-Fi.

Finally, in order to investigate the practical issues of deploying our information processing algorithm on mobile computers or PDAs that will communicate directly with the sensors constituting a pervasive network, we are developing prototype stand-alone weather sensors that will be deployed at the University of Southampton (see figure 7). These sensors incorporate Wi-Fi web servers and make their readings available in the same RDF format described in Section 4.

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

[1] Microsoft SenseWeb project. See `http://research.microsoft.com/nec/senseweb/`.

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

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

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

[5] A. Deshpande, C. Guestrin, S. Madden, J. Hellerstein, and W. Hong. Model-driven data acquisition in sensor networks.

In *Proceedings of the Thirtieth International Conference on Very Large Data Bases (VLDB 2004)*, pages 588–599, 2004.

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

[7] J. K. Hart and K. Martinez. Environmental Sensor Networks: A revolution in the earth system science? *Earth-Science Reviews*, 78:177–191, 2006.

[8] A. Girard, C. Rasmussen, J. Candela, and R. Murray-Smith. 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, 2003.

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

[10] A. Krause, C. Guestrin, A. Gupta, and J. Kleinberg. Near-optimal sensor placements: Maximizing information while minimizing communication cost. In *Proceedings of the Fifth International Conference on Information Processing in Sensor Networks (IPSN '06)*, pages 2–10, Nashville, Tennessee, USA, 2006.

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

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

[13] R. Murty, A. Gosain, M. Tierney, A. Brody, A. Fahad, J. Bers, and M. Welsh. Citysense: A vision for an urban-scale wireless networking testbed. Technical Report TR-13-07, Harvard University, September 2007.

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

[15] M.A. Osborne and S. J. Roberts. Gaussian processes for prediction. Technical Report PARG-07-01. Available at `www.robots.ox.ac.uk/~parg/publications.html`, University of Oxford, September 2007.

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

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

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

[19] M. Sasena. *Flexibility and Efficiency Enhancements for Constrained Global Design Optimization with Kriging Approximations*. PhD thesis, University of Michigan, 2002.

[20] M. Stein. Space-Time Covariance Functions. *Journal of the American Statistical Association*, 100(469):310–322, 2005.

[21] B. Szekely and E. Torres. A semantic data collection model for sensor network applications. Available at `http://www.klinewoods.com/papers/semanticdcn.pdf`.

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

[23] The MathWorks. MATLAB R2007a, 2007. 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 $\boldsymbol{x} = \mathbf{A} \backslash \boldsymbol{b}$ 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:

$$S_{1,1} = R_{1,1} \tag{16}$$

$$S_{1,2} = R_{1,1}^{\mathrm{T}} \setminus V_{1,2} \tag{17}$$

$$S_{1,3} = R_{1,3} \tag{18}$$

$$S_{2,2} = \mathrm{chol}(V_{2,2} - S_{1,2}^{\mathrm{T}} S_{1,2}) \tag{19}$$

$$S_{2,3} = S_{2,2}^{\mathrm{T}} \setminus (V_{2,3} - S_{1,2}^{\mathrm{T}} S_{1,3}) \tag{20}$$

$$S_{3,3} = \mathrm{chol}(R_{3,3}^{\mathrm{T}} R_{3,3} - S_{2,3}^{\mathrm{T}} S_{2,3}) \tag{21}$$

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 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:

$$R_{1,1} = S_{1,1} \tag{22}$$

$$R_{1,3} = S_{1,3} \tag{23}$$

$$R_{3,3} = \mathrm{chol}(S_{2,3}^{\mathrm{T}} S_{2,3} + S_{3,3}^{\mathrm{T}} S_{3,3}) \tag{24}$$

Note that the special structure of equation (24) can be exploited for the efficient resolution of the required Cholesky operation, as, for example, in the MATLAB function cholupdate [23]. 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.