Gaussian Processes for Active Data Mining of Spatial Aggregates

Naren Ramakrishnan[†], Chris Bailey-Kellogg[#], Satish Tadepalli[†], and Varun N. Pandey[†]

[†]Department of Computer Science, Virginia Tech, Blacksburg, VA 24061
[#]Department of Computer Science, Dartmouth College, Hanover, NH 03755

Abstract

Active data mining is becoming prevalent in applications requiring focused sampling of data relevant to a high-level mining objective. It is especially pertinent in scientific and engineering applications where we seek to characterize a configuration space or design space in terms of spatial aggregates, and where data collection can become costly. Examples abound in domains such as aircraft design, wireless system simulation, fluid dynamics, and sensor networks. This paper develops an active mining mechanism, using Gaussian processes, for uncovering spatial aggregates from only a sparse set of targeted samples. Gaussian processes provide a unifying framework for building surrogate models from sparse data, reasoning about the uncertainty of estimation at unsampled points, and formulating objective criteria for closing-the-loop between data collection and data mining. Our mechanism optimizes sample selection using entropy-based functionals defined over spatial aggregates instead of the traditional approach of sampling to minimize estimated variance. We apply this mechanism on a global optimization benchmark comprising a testbank of 2D functions, as well as on data from wireless system simulations. The results reveal that the proposed sampling strategy makes more judicious use of data points by selecting locations that clarify high-level structures in data, rather than choosing points that merely improve quality of function approximation.

Keywords: spatial mining, active mining, sparse data, spatial aggregation, Gaussian processes.

1 Introduction

Many data mining applications in scientific and engineering contexts require analysis and mining of spatial datasets derived from computer simulations or field data, e.g., wireless system simulations, aircraft design configuration spaces, fluid dynamics simulations, and sensor network optimization. In contrast to traditional data mining contexts that are dominated by massive datasets, these domains are actually characterized by a *paucity* of data, owing to the cost and time involved

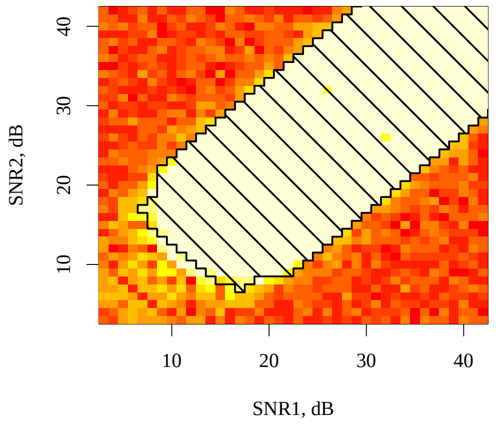


Figure 1: Mining configuration spaces from wireless system simulations. The shaded region denotes the largest portion of the configuration space where we can claim, with confidence at least 99%, that the average bit error rate (BER) is acceptable for voice-based system usage. Each 'cell' in the plot is the result of the spatial and temporal aggregation of hundreds of wireless system simulations, many of which take hours.

in conducting simulations or setting up experimental apparatus for data collection. Nevertheless, the computational scientist has control of *where* data can be collected; it is hence prudent in such domains to focus data collection in only those regions that are deemed important to support a high-level data mining objective.

As a concrete example, consider the characterization of WCDMA (wideband code-division multiple access) wireless system configurations for a given indoor environment. A configuration comprises many adjustable parameters, and the goal of wireless system characterization is to assess the relationship between these parameters and performance metrics such as BER (bit error rate), a measure of the number of bits transmitted in error using the system. When a wireless engineer designs a system for a given indoor environment, he or she sets an acceptable performance criterion for BER (e.g., 10^{-3} for a system designed to carry voice traffic, stricter thresholds for data traffic) and seeks a region in the configuration space that can satisfy this criterion (see Fig. 1). To collect the data necessary for mining configuration spaces, the engineer either performs a costly Monte Carlo simulation (where a model of radio propagation in the wireless channel is embedded inside a system-wide model encapsulating wireless protocols and communications standards), or installs channel sounding equipment and system instrumentation in the environment, and actually enacts usage scenarios. In either approach, it is not feasible to first organize a voluminous body of data and subsequently perform data mining on the collected dataset. It is thus imperative that we interleave data collection and data mining and focus sampling at only those locations that maximize well-defined notions of relevance and utility. Importantly, we will not need to sample the entire configuration space, only enough so as to identify a region with acceptable confidence.

Active data selection has been investigated in a variety of contexts [4, 25]. A sampling strategy typically embodies a human assessment of where might be a good location to collect data [1, 13] or is derived from the optimization of specific design criteria [5, 17, 22]. Many of these strategies, however, are either based on utility of data for function approximation purposes [24], or are meant to be used with specific data mining algorithms and tasks (e.g., classification [10]). In this paper, we present a formal framework that casts spatial data mining as uncovering successive multi-level aggregates of data, and uses properties of higher-level structures to help close the loop between mining and data collection. This approach helps us design sampling strategies that bridge higher-level quality metrics of structures (e.g., entropy) with lower-level considerations of data samples (e.g., locations and fidelity). While we focus on spatial contexts, we point out that *spatial* can denote any dimension that affords a metric; our approach thus applies equally well to a wide range of data sets with more abstract notions of space (such as the wireless simulation example above).

Our active mining mechanism is based on the spatial aggregation language (SAL; [3]), a generic data mining framework for spatial datasets, and Gaussian processes (GPs; [27]), a powerful unifying theory for approximating and reasoning about datasets. Gaussian processes provide the 'glue' that enables us to perform active mining on spatial aggregates. In particular, they aid in (i) creation of *surrogate models* from data using a sparse set of samples (for cheap generation of dense approximate datasets), (ii) reasoning about the uncertainty of estimation at unsampled points, and (iii) formulation of objective criteria for active data collection.

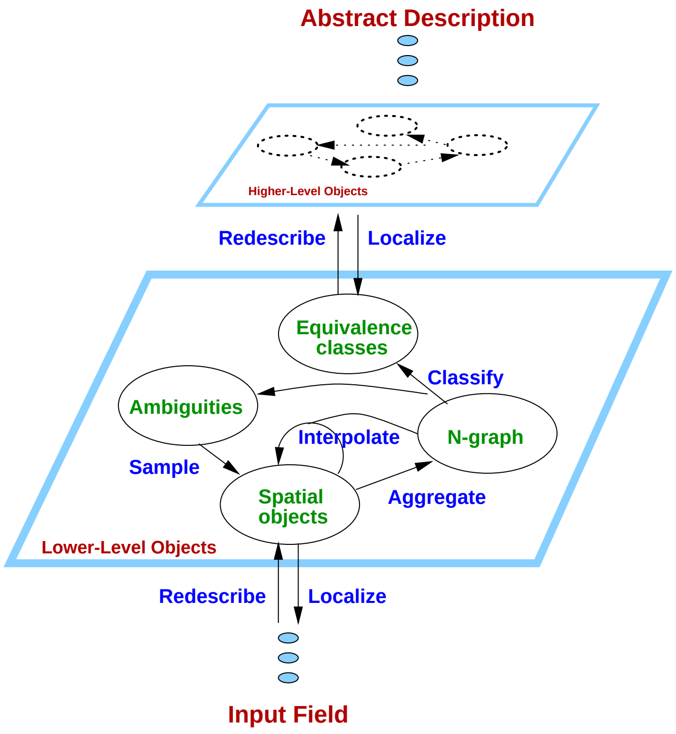


Figure 2: SAL uncovers multi-layer spatial aggregates by employing a small set of operators (a spatial mining "vocabulary") utilizing suitable domain knowledge. A variety of spatial data mining tasks, such as vector field bundling, contour aggregation, correspondence abstraction, clustering, and uncovering regions of uniformity can be expressed as multi-level spatial aggregate computations.

1.1 Contributions: This paper builds on our prior work in [1, 23] by presenting a novel integration of Gaussian processes with SAL:

- While classical active mining work in spatial modeling focuses on quality of function approximation, the mechanism presented here focuses on clarifying high-level structures. The entropy-based sampling approach introduced in this paper is applicable to mining a broad range of spatial structures.

- Unlike traditional data mining contexts that deal with voluminous amounts of data, the mechanism is targeted at scenarios where data collection costs far outshadow data mining costs. For instance, in the wireless simulation study, each data sample requires a few hours to acquire on a cluster of workstations whereas the data mining (and sample selection optimization) algorithms as implemented here can be executed in a matter of minutes on a workstation.

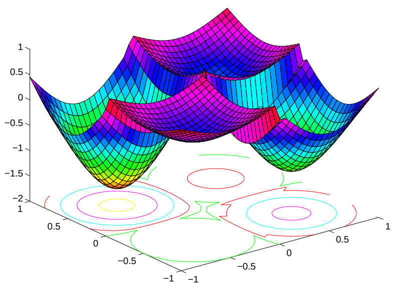


Figure 3: de Boor's 'pocket' function in 2D, depicting contours around basins of local minima.

- Since Gaussian processes are re-statements of kernel-based learning methods [7] this work helps bridge the qualitative nature of SAL algorithms with rigorous quantitative methodologies necessary to evaluate and assess active mining strategies.

This work assumes a moderate background of algorithms for spatial aggregation and spatial statistical modeling. Nevertheless, Sec. 2 and Sec. 3 overview earlier work on SAL and GPs for the SIAM DM audience and instantiate such work for a motivating case study — identifying and characterizing pockets in a space (such as the wireless system design configuration space). Sec. 4 then moves to active mining and introduces two important classes of sampling strategies integrating SAL and GPs. Sec. 5 evaluates the mechanism using both synthetic and real-world datasets. Sec. 6 provides a discussion and reviews related work.

2 Spatial Aggregation Language

The Spatial Aggregation Language (SAL) [3, 28, 30] is a generic framework to study the design and implementation of spatial data mining algorithms. SAL is centered on a *field ontology*, in which the spatial data input is a field mapping from one continuum to another (e.g. 2-D temperature field: $\mathbf{R}^2 \to \mathbf{R}^1$; 3-D fluid flow field: $\mathbf{R}^3 \to \mathbf{R}^3$). SAL programs employ vision-like routines, in an *imagistic reasoning* style [29], to uncover and manipulate multi-layer geometric and topological structures in fields. Due to continuity, fields exhibit regions of uniformity, and these regions can be abstracted as higher-level structures which in turn exhibit their own continuities. Task-specific domain knowledge specifies how to uncover such uniformity, defining metrics for closeness of objects and similarity of features. For example, streamlines are connected curves of nearby points with vectors flowing in similar enough directions, while pressure cells are connected regions of similar (and extreme) enough pressure.

SAL supports structure discovery through a small set of generic operators, parameterized with domain-specific knowledge, on uniform data types. These operators and data types mediate increasingly abstract descriptions of the input data (see Fig. 2) to form higher-level abstractions and mine patterns. The *primitives* in SAL are contiguous regions of space called *spatial objects*; the *compounds* are (possibly structured) collections of spatial objects; the *abstraction mechanisms* connect collections at one level of abstraction with single objects at a higher level. This vocabulary has proved effective for expressing the mechanisms required to uncover multi-level structures in spatial datasets in applications ranging from decentralized control design [2] and object manipulation [30] to analysis of weather data [12], diffusion-reaction morphogenesis [21], and matrix perturbation analysis [22].

The identification of structures in a field is a form of data reduction: a relatively information-rich field representation is abstracted into a more concise structural representation (e.g. pressure data points into isobar curves or pressure cells; isobar curve segments into troughs). Navigating the mapping from field to abstract description through multiple layers rather than in one giant step allows the construction of modular data mining programs with manageable pieces that can use similar processing techniques at different levels of abstraction. The multi-level mapping also allows higher-level layers to use global properties of lower-level objects as local properties of the higher-level objects. For example, the average temperature in a region is a global property when considered with respect to the temperature data points, but a local property when considered with respect to a more abstract region description. As this paper demonstrates, analysis of higher-level structures in such a hierarchy can guide interpretation of lower-level data.

Let us begin with a spatial mining task motivated by the wireless study — determining the number and locations of *pockets*, or basins of local minima, in a vector field. Fig. 3 illustrates four pockets in a field defined by Carl de Boor's function in 2D (from [22]):

$$\begin{aligned}
&(2.1)\ \alpha(\mathbf{X}) &=& \cos\left(\sum_{i=1}^{n} 2^i \left(1 + \frac{x_i}{\mid x_i \mid}\right)\right) - 2 \\
&(2.2)\ \delta(\mathbf{X}) &=& \|\mathbf{X} - 0.5\mathbf{I}\| \\
&(2.3)\ p(\mathbf{X}) &=& \alpha(\mathbf{X})(1 - \delta^2(\mathbf{X})(3 - 2\delta(\mathbf{X}))) + 1
\end{aligned}$$

where $\mathbf{X}$ is the n-dimensional point $(x_1, x_2, \cdots, x_n)$ at which the pocket function p is evaluated, $\mathbf{I}$ is the identity n-vector, and $\| \cdot \|$ is the L_2 norm. The property of this function is that it assumes a pocket in each corner of the cube, just outside the sphere enclosed in the cube. Since the ratio of cube volume (2^n) to that of the sphere ($\pi^{n/2}/(n/2)!$) grows unboundedly, global optimization algorithms cannot exploit any special properties and must consider every one of the 2^n corners! Hence, the de Boor function is a well-known benchmark for global optimization (esp. in high dimensions), but we focus here on a somewhat different objective of characterizing the high-level structure of the field. The algorithmic encoding of the calculus definition of local minima suggests that the four pockets in Fig. 3 can be identified via convergent flows in the gradient underlying the vector field. Let us assume we are given a dense set of samples covering the region of interest. Fig. 4 illustrates an example of key spatial aggregation operations:

(a) Establish the input field, here by calculating the gradient field (normalized, since we're interested only in direction in order to detect convergence).

(b) Localize computation with a *neighborhood graph*, so that only spatially proximate objects are compared. Here, an 8-adjacency neighborhood graph is employed, which results in somewhat 'blocky' streamlines but fast computation.

(c)-(f) Use a series of local computations to find *equivalence classes* of neighboring objects with similar features. Here, we systematically eliminate all neighborhood graph edges but those whose directions best match the vector direction at both endpoints. 'Forward neighbor' computation compares graph edge direction with the average of the vector directions, and keeps only those that are similar enough (implemented as a cosine angle similarity threshold). 'Best forward neighbor' at junction points then selects from among these neighbors, by a third metric combining similarity in direction with closeness in point location. Backward calculations are analogous, but deal with the predecessor along a streamline rather than the successor.

(g) Move up a level in the spatial object hierarchy by *redescribing* equivalence classes into more abstract objects. Here, connected vectors are abstracted into curve objects, which have both a reduced representation and additional semantic properties (e.g. curvature is well-defined).

(h) Apply the same mechanism — aggregate, classify, and redescribe — at the new level, *using the exact same operators but with different metrics.* Here, curves are grouped into coherent pockets with convergent flow. Neighborhood (not shown) is derived from neighborhood of constituent vectors, and equivalence tests direction of flow for convergence.

Notice that SAL is not a specific data mining algorithm, but rather a language to construct complex mining operations (such as in Fig. 4) from a small core set of operations. As such, the quality of results from a SAL implementation depends on suitable choices of abstraction levels and appropriate settings of any relevant parameters. For instance, in the above example, three parameters control the relationship from input field to output structures: adjacency neighborhood size (used in step (b)), angle for vector similarity (used in step (c)), and distance penalty metric (used in step (d) to combine distance with direction). For Fig. 4, we set these parameters to 1.5 (generates an 8-adjacency neighborhood), 0.75, and 0.1 respectively. This paper is not concerned with evaluating particular SAL implementations but instead focuses on using them from within an active mining context.

Localized computations are integral to SAL, and hence an effective SAL application relies on a dense set of samples covering the domain. When data is scarce, we can first build an approximation to the underlying field with the given samples, and use the approximation to generate a dense field of data (e.g., on a uniform grid). Such an approximation is called a *surrogate* model — cheap-to-compute substitutes for complex functions. One way to build surrogate models relies on Gaussian processes.

3 Gaussian Processes

The use of Gaussian processes in machine learning and data mining is a relatively new development, although their origins can be traced to spatial statistics and the practice of modeling known as kriging [14]. In contrast to global approximation techniques such as least-squares fitting, GPs are local approximation techniques, akin to nearest-neighbor procedures. In contrast to function approximation techniques that place a prior on the form of the function, GP modeling techniques place a prior on *the covariance structures* underlying the data.

The basic idea in GPs is to model a given dataset as a realization of a stochastic process. Formally, a GP is a set of random variables any finite subset of which have a (multivariate) normal distribution. For our purposes, we can think of these variables as spatially distributed (scalar) response variables t_i, one for each 2D location $\mathbf{x_i} = [x_{i1}, x_{i2}]$ where we have collected a data sample. In our vector field analysis application, t_i denotes the modeled response, i.e., the value of de Boor's function at $\mathbf{x_i}$. Given a dataset $\mathcal{D} = \{\mathbf{x_i}, t_i\}, i = 1 \ldots n$, and a new data point $\mathbf{x_{n+1}}$, a GP can be used to model the posterior $P(t_{n+1}|\mathcal{D}, x_{n+1})$ (which would also be a Gaussian). This is essentially what many Bayesian modeling techniques do (e.g., least squares

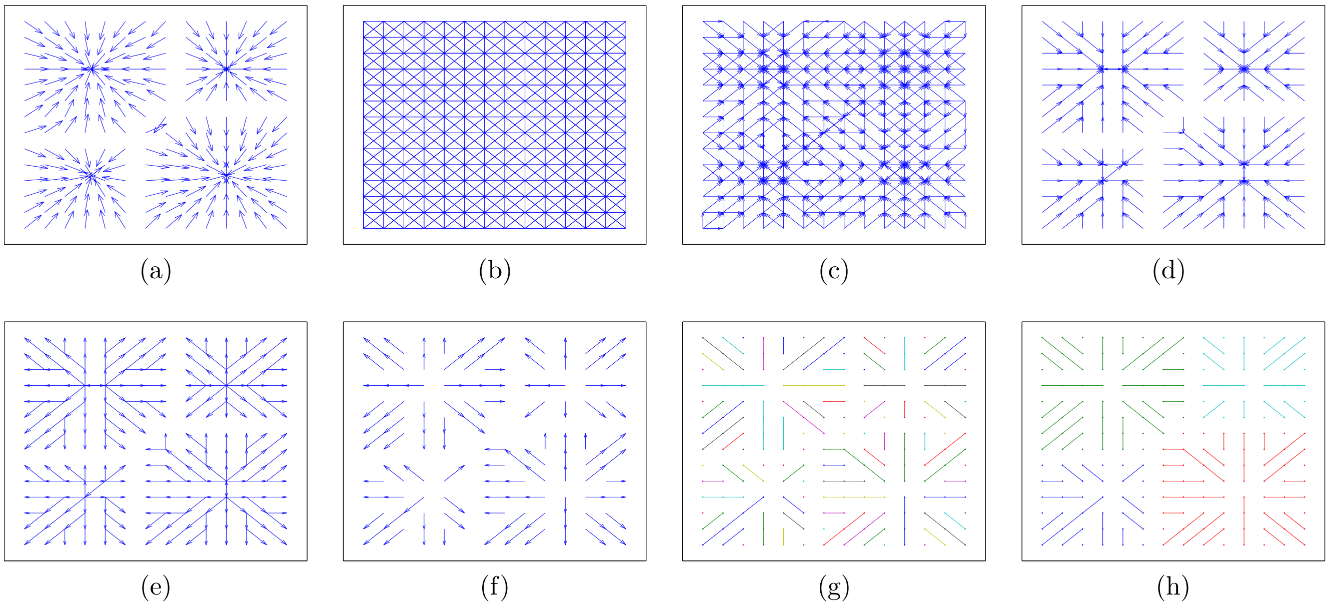


Figure 4: Example steps in SAL implementation of vector field analysis of de Boor's function. (a) Input vector field. (b) 8-adjacency neighborhood graph. (c) Forward neighbors. (d) Best forward neighbors. (e) Neighborhood graph transposed from best forward neighbors. (f) Best backward neighbors. (g) Resulting adjacencies redescribed as curves. (h) Higher-level aggregation and classification of curves whose flows converge.

approximation with normally distributed noise) but it is the specifics of how the posterior is modeled that make GPs distinct as a class of modeling techniques.

To make a prediction of t_{n+1} at a point $\mathbf{x_{n+1}}$, GPs place greater reliance on t_i's from nearby points. This reliance is specified in the form of a covariance prior for the process and will be central to how we embed SAL in a broader GP framework:

$$\text{(3.4)}\ \mathrm{Cov}(t_i, t_j) = \alpha \ \exp\left(-\frac{1}{2}\sum_{k=1}^{2} a_k (x_{ik} - x_{jk})^2\right)$$

Intuitively, this function captures the notion that response variables at nearby points must have high correlation. The reader will note that this idea of influence decaying with distance has an immediate parallel to how SAL programs localize computations. In Eq. 3.4, α is an overall scaling term whereas a_1, a_2 define the length scales for the two dimensions. However, this prior (or even its posterior) does not directly allow us to determine t_j from t_i, since the structure only captures the covariance; predictions of a response variable for new sample locations are thus conditionally dependent on the measured response variables and *their* sample locations. Hence, we must first estimate the covariance parameters (a_1, a_2, and α) from $\mathcal{D}$ and then use these parameters *along with* $\mathcal{D}$ to predict t_{n+1} at $\mathbf{x_{n+1}}$.

3.1 Using a GP: Before covering the learning procedure for the covariance parameters (a_1, a_2, and α), it is helpful to develop expressions for the posterior of the response variable in terms of these parameters. Since the jpdf of the response variables $P(t_1, t_2, \cdots, t_{n+1})$ is modeled Gaussian (we will assume a mean of zero), we can write:

$$P(t_1, t_2, \cdots, t_{n+1} \,|\, \mathbf{x_1}, \mathbf{x_2}, \cdots, \mathbf{x_{n+1}}, \mathrm{Cov}_{n+1}) = \frac{1}{\lambda_1} \exp\left(-\frac{1}{2}\,[t_1, t_2, \cdots, t_{n+1}]\ \mathrm{Cov}_{n+1}^{-1}\ [t_1, t_2, \cdots, t_{n+1}]^T\right)$$

where we ignore λ_1 as it is simply a normalizing factor. Here, Cov_{n+1} is the covariance matrix formed from the $(n+1)$ data values $(\mathbf{x_1}, \mathbf{x_2}, \cdots, \mathbf{x_{n+1}})$. A distribution for the unknown variable t_{n+1} can then be obtained as:

$$\begin{aligned} P(t_{n+1} | t_1, t_2, \cdots, t_n, \mathbf{x_1}, \mathbf{x_2}, \cdots, \mathbf{x_{n+1}}, \mathrm{Cov}_{n+1}) &= \frac{P(t_1, t_2, \cdots, t_{n+1} \,|\, \mathbf{x_1}, \mathbf{x_2}, \cdots, \mathbf{x_{n+1}}, \mathrm{Cov}_{n+1})}{P(t_1, t_2, \cdots, t_n \,|\, \mathbf{x_1}, \mathbf{x_2}, \cdots, \mathbf{x_{n+1}}, \mathrm{Cov}_{n+1})} \\ &= \frac{P(t_1, t_2, \cdots, t_{n+1} \,|\, \mathbf{x_1}, \mathbf{x_2}, \cdots, \mathbf{x_{n+1}}, \mathrm{Cov}_{n+1})}{P(t_1, t_2, \cdots, t_n \,|\, \mathbf{x_1}, \mathbf{x_2}, \cdots, \mathbf{x_n}, \mathrm{Cov}_n)} \end{aligned}$$

where the last step follows by conditional independence of $\{t_1, t_2, \cdots, t_n\}$ w.r.t. $\mathbf{x_{n+1}}$ and the part of Cov_{n+1} not contained in Cov_n. The denominator in the above expression is another Gaussian random variable, given by:

$$P(t_1, t_2, \cdots, t_n \,|\, \mathbf{x_1}, \mathbf{x_2}, \cdots, \mathbf{x_n}, \mathrm{Cov}_n) = \frac{1}{\lambda_2} \exp\left(-\frac{1}{2}\,[t_1, t_2, \cdots, t_n]\ \mathrm{Cov}_n^{-1}\ [t_1, t_2, \cdots, t_n]^T\right)$$

Putting it all together, we get:

$$P(t_{n+1}|t_1, t_2, \cdots, t_n, \mathbf{x_1}, \mathbf{x_2}, \cdots, \mathbf{x_{n+1}}, \mathrm{Cov}_{n+1}) = \frac{\lambda_2}{\lambda_1} \exp\Big(-\frac{1}{2}[t_1, t_2, \cdots, t_{n+1}]\ \mathrm{Cov}_{n+1}^{-1}\ [t_1, t_2, \cdots, t_{n+1}]^T - \frac{1}{2}[t_1, t_2, \cdots, t_n]\ \mathrm{Cov}_n^{-1}\ [t_1, t_2, \cdots, t_n]^T \Big)$$

Computing the mean and variance of this Gaussian distribution, we get an estimate of t_{n+1} as:

$$\hat{t}_{n+1} = \mathbf{k}^T \mathrm{Cov}_n^{-1} [t_1, t_2, \cdots, t_n] \tag{3.5}$$

and our uncertainty in this estimate as:

$$\sigma^2_{\hat{t}_{n+1}} = k - \mathbf{k}^T \mathrm{Cov}_n^{-1} \mathbf{k} \tag{3.6}$$

where $\mathbf{k}^T$ represents the n-vector of covariances with the new data point:

$$\mathbf{k}^T = [\mathrm{Cov}(\mathbf{x_1}, \mathbf{x_{n+1}})\ \mathrm{Cov}(\mathbf{x_2}, \mathbf{x_{n+1}}) \cdots \mathrm{Cov}(\mathbf{x_n}, \mathbf{x_{n+1}})]$$

and k is the $(n+1, n+1)$ entry of Cov_{n+1}. Eqs. 3.5 and 3.6, together, give us both an approximation at any given point and an uncertainty in this approximation; they will serve as the basic building blocks for closing-the-loop between data modeling and higher level mining functionality.

The above expressions can be alternatively derived by positing a linear probabilistic model and optimizing for the MSE (mean squared error) between observed and predicted response values (e.g., see [24]). In this sense, the Gaussian process model considered here is also known as the BLUE (best linear unbiased estimator), but GPs are not restricted to linear combinations of basis functions.

To apply GP modeling to a given dataset, we must first ensure that the chosen covariance structure matches the data characteristics. We have chosen a stationary structure above under the assumption that the covariance is translation invariant. Various other functions have been studied in the literature (e.g., see [18, 19, 24]), all of which satisfy the required property of positive definiteness. The simplest covariance function yields a diagonal matrix, but this means that no data sample can have an influence on other locations, and the GP approach offers no particular advantages. In general, by placing a prior directly on the function space, GPs are appropriate for modeling 'smooth' functions. The terms a_1, a_2 capture how quickly the influence of a data sample decays in each direction and, thus, the length scales for smoothness.

An important point to note is that even though the GP realization is one of a random process, we can nevertheless build a GP model for deterministic functions (like the de Boor's function) by choosing a covariance structure that ensures the diagonal correlations to be 1 (i.e., perfect reproducibility when queried for a sample whose value is known). Also, the assumption of zero mean for the Gaussian process can be relaxed, by including a constant term (gives another parameter to be estimated) in the covariance formulation. This approach is used for our experimental studies.

3.2 Learning a GP: Learning the GP parameters $\theta = (a_1, a_2, \alpha)$ can be undertaken in the ML and MAP frameworks, or in the true Bayesian setting where we obtain a distribution over values. The log-likelihood for the parameters is given by:

$$\begin{aligned} \mathcal{L} &= \log P(t_1, t_2, \cdots, t_n | \mathbf{x_1}, \mathbf{x_2}, \cdots, \mathbf{x_n}, \theta) \\ &= c + \log P(\theta) - \frac{n}{2}\log(2\pi) - \frac{1}{2}\log \mid \mathrm{Cov}_n \mid \\ &\quad -\frac{1}{2}[t_1, t_2, \cdots, t_n]\ \mathrm{Cov}_n^{-1}\ [t_1, t_2, \cdots, t_n]^T \end{aligned}$$

To optimize for the parameters, we can compute partial derivatives of the log-likelihood for use with a conjugate gradient or other optimization algorithm:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \theta} &= \frac{\partial \log P(\theta)}{\partial \theta} \\ &\quad -\frac{1}{2}\ \mathrm{tr}\ \left(\mathrm{Cov}_n^{-1}\ \frac{\partial\, \mathrm{Cov}_n^{-1}}{\partial \theta} \right) \\ &\quad +\frac{1}{2}\ [t_1, t_2, \cdots, t_n]\ \mathrm{Cov}_n^{-1}\ \frac{\partial\, \mathrm{Cov}_n^{-1}}{\partial \theta} \\ &\qquad \mathrm{Cov}_n^{-1}\ [t_1, t_2, \cdots, t_n]^T \end{aligned}$$

where $\mathrm{tr}(\cdot)$ denotes the trace function. In our running example, we need only estimate three parameters for θ, well within the purview of modern numerical optimization software. For larger numbers of parameters, we can resort to the use of MCMC methods [19].

4 Active Data Mining Strategies

The above section showed two important uses of GPs for spatial mining: designing a surrogate function for generating a dense field (via Eq. 3.5), and assessing uncertainties in our estimates of the function at unsampled points (using Eq. 3.6). We are now ready to formulate objective criteria for active data selection, a pre-cursor to active mining.

4.1 Variance Reducing Designs: A simple strategy for sampling is to target locations to reduce our uncertainty in modeling, i.e., select the location that minimizes the posterior generalized variance of the function. This approach can be seen as optimizing sample selection for the functional:

$$\Phi_V = \frac{1}{2} \log \left[\frac{\partial t}{\partial \theta}\right] \mathcal{H}^{-1} \left[\frac{\partial t}{\partial \theta}\right]^T \tag{4.7}$$

where $\left[\frac{\partial t}{\partial \theta}\right]$ is the (row) vector of sensitivities w.r.t. each GP parameter computed at a sample location, and $\mathcal{H}$ is the Hessian (second order partial derivatives) of t, again w.r.t. the parameters. A straightforward derivation will show that optimizing Φ_V suggests a location whose 'error bars' σ^2 are highest.

To implement this strategy, we can adopt either a block design (optimize for K locations simultaneously), or apply it sequentially to determine one extra sampling location at a time. The former is appropriate when we can farm out function evaluations across nodes in a compute cluster, whereas the latter will track the design functional better. We adopt the sequential approach here; Fig. 5 shows this strategy for the pocket function of Fig. 3 and concomitant results from pocket mining of the surrogate model data. At each step, we determine the best sample location (from among unsampled locations on a regular grid of 21×21), build the GP model from the data collected thus far, and apply our SAL-based vector aggregation mechanism to the gradient field derived from the function values.

The initial design has one point in the center of each quadrant, and one at the center. Not surprisingly, we find a significant number (16) of basins in the gradient field. The next four points added are actually at the corners; this is because estimated variances are typically high toward the boundaries of an interpolation region. As MacKay points out [17], such a metric has a tendency to 'repeatedly gather data at the edges of the input space.' Continuing the sampling, we see that the 13-point design actually has the samples organized in a diagonal design (a layout that has been referred to as 'whimsical' [9]). The emphasis on overall quality of function approximation more than data mining is evident from the fact that it takes over 30 points before the SAL-based pocket finder can infer that there are four pockets. In further experiments not reported here, we have found that pushing the initial points outward (or inward) does not have any appreciable effect on future samplings, and the variance-based metric favors the outer envelope of the design space.

4.2 Entropy-Based Functionals:

It is a classical result in experiment design (e.g., see [8]) that, for Gaussian priors, the variance-reducing design is actually equivalent to the design minimizing the (expected) posterior entropy of the distribution $t_{\overline{\mathcal{D}}|\mathcal{D}}$, where $\overline{\mathcal{D}}$ denotes the unsampled locations in $\mathcal{D}$. For a proof, see [16]. This criterion is also equivalent to the D-optimality design criterion in spatial statistics, under the assumption that the noise factor on all measurements is the same. MacKay generalizes this idea [17], and pre-specifies a collection of points requiring high-quality approximation; the goal then is to minimize entropy of data distribution w.r.t. these points. This strategy does not apply here since our understanding of which locations are relevant improves as active mining proceeds.

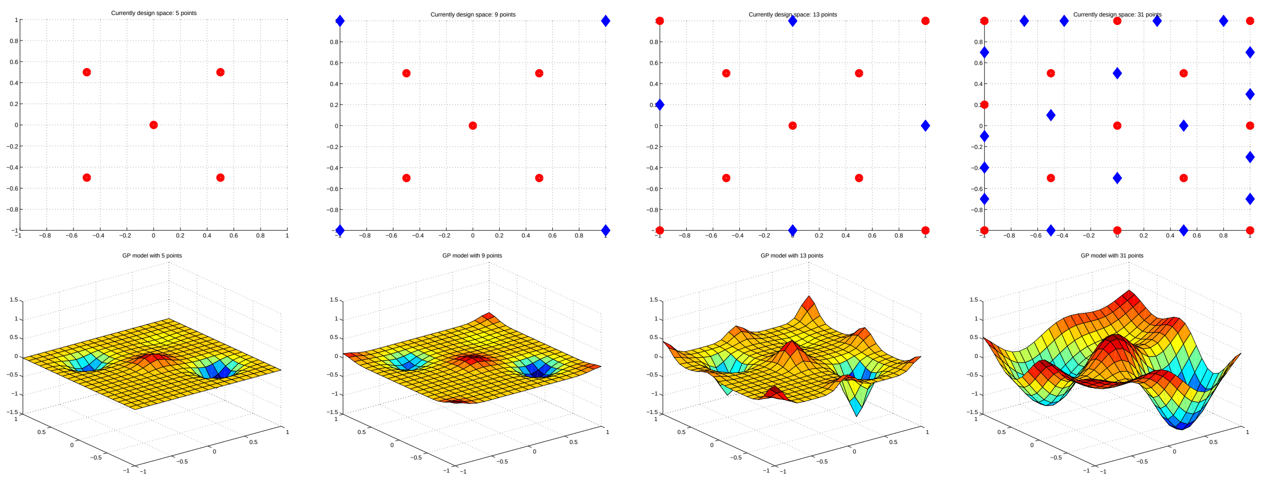
To develop a better active mining strategy, notice that our goal is the identification of regions defined by convergent flows. If we view the SAL program as an information processor that maps a data field into a class field (defined over the same underlying space), then the utility of sampling in a region is directly related to our inferential capabilities about the corresponding region in the class field. Intuitively, we should be more interested in samples that tell us something about the boundary between regions than those that capture the insides of a region, *even though the latter might have high variance in its current estimate*. Repeatedly sampling function values inside an already classified and abstracted region is not as useful as sampling to clarify an emerging boundary classification. This means that we must bridge high-level information about pockets from SAL into a preference of where to collect data.

An idea that suggests itself is to adopt variance-based design, but instead of minimizing the entropy of the data distribution, minimize the entropy of the class distribution as revealed by the SAL pocket finder. By positing a class distribution at each point, based on the class labels occupied by neighboring points, we achieve our goal of ranking locations along region boundaries higher. While this basic strategy appears reasonable, it will repeatedly gather information at the region boundaries, just as variance-based design repeatedly focuses on the edges. So a point with high entropy is a good location to sample only as long as the variance surrounding it is sufficiently high. As our confidence in the data value increases, our preference for this location should decrease even if the class entropy remains large (as it will, if it lies on a boundary). This suggests using class entropy to define a distribution $P_E(\mathbf{x})$ over points, and using that distribution to scale the variance-based design criterion:

$$\Phi_E = \frac{1}{2} \sum_{\mathbf{x}} P_E(\mathbf{x}) \ \log \left[\frac{\partial t}{\partial \theta}\right] \mathcal{H}^{-1} \left[\frac{\partial t}{\partial \theta}\right]^T \tag{4.8}$$

The expression inside the summation contains the same term as in Eq. 4.7 but is now evaluated across the design space and scaled by the amount of interest in location $\mathbf{x}$:

$$P_E(\mathbf{x_i}) \propto \sum_{\mathbf{x} \in \mathcal{N}(\mathbf{x_i})} P(C(\mathbf{x})) \ \log P(C(\mathbf{x})) \tag{4.9}$$



16	**3**	**3**	**4**

Figure 5: How variance-based sampling selects locations. (top row) initial design of 5 points, followed by snapshots taken at later stages (9, 13, and 31 points). Old sample locations are shown with red circles and new locations are shown with blue diamonds. (middle row) GP model fits to the given samples. (bottom row) Number of pockets identified by SAL pocket miner.

where $\mathcal{N}(\mathbf{x_i})$ is a neighborhood around $\mathbf{x_i}$, $C(\mathbf{x})$ denotes the (flow) class of point $\mathbf{x}$ as inferred by the SAL miner, and $P(C(\mathbf{x}))$ denotes the probability of encountering this class in the neighborhood. The proportionality constant in Eq. 4.9 must be set to ensure that $\sum P_E = 1$. Formal characterization of this criterion (i.e., convergence properties) is difficult since $P_E(\mathbf{x})$ changes during every iteration of data mining, and we do not have a model of how $P_E(\mathbf{x})$ varies across samplings. Operationally, to apply this criterion, we can identify the location that gives the highest information *gain*, given that we are intending to make a measurement at that location. Fig. 6 shows a design that optimizes Φ_E and successfully reveals all four pockets with only 11 points.

4.3 Computational Considerations: Other than any data collection costs, the primary costs to implementing the active mining mechanisms involve the nested optimizations and the necessary matrix computations. There are two optimizations per round of data collection: a multi-dimensional optimization over θ to fit the surrogate model, and a 2D optimization over $\mathbf{x}$ to identify the next sample point. Both can be done either locally or globally, depending on our fidelity requirements and availability of resources. Here, to reduce the computational complexity in building the surrogate model, we adopt the public domain Netlab scaled conjugate gradient algorithm [18] which runs in $O(|\mathcal{D}||\theta|)$ time. While this algorithm avoids having to work with the Hessian explicitly, the active sample selection step requires the computation of the Hessian inverse, which

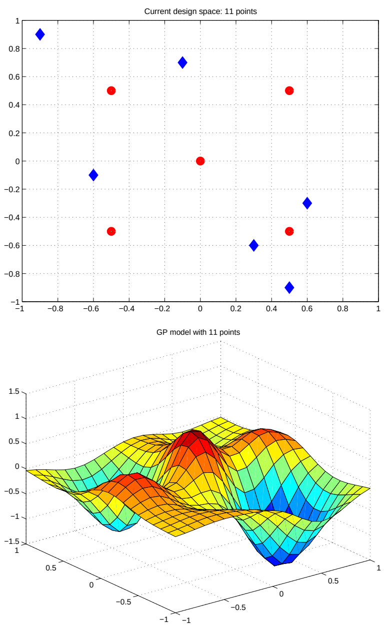


Figure 6: Active data mining with an entropy-based functional. This sampling strategy picks six additional points (top), in various quadrants; the SAL miner finds four pockets (bottom) when a GP model is constructed using the given points. Contrast this with the performance of variance-based sampling for a comparable number of samples.

takes $O(|\mathcal{D}||\theta|^2 + |\theta|^3)$ time. To reduce the cost of optimization, we use a discrete lattice search or hill climbing, restricting our attention to locations over a uniform grid. If the number of locations on the grid is $|G|$, then each round of active mining requires $O(|G||\theta|^2)$ time, plus the cost of computing the inverse Hessian. This expression applies to both variance-based mining and entropy-based mining, since the computation of $P_E(\mathbf{x})$ is just linear in $|G|$ for a fixed neighborhood calculation of entropy. Recall that this cost is typically negligible compared to the actual cost of running a simulation (to acquire a data sample).

4.4 Stopping Criteria: How do we know when to stop sampling? If a cost metric is defined over data collection, and if it can be determined that we can sample at most K points within the given resources, then we should ideally perform a K-dimensional optimization, rather than adopting a sequential sampling strategy. In the absence of such a cost-metric, a sampling strategy could terminate when the estimated dataset log likelihood is within bounds. In this paper, we primarily evaluate sampling strategies using classes of problems for which the 'right' answer is known, and pose questions such as: 'starting from an initial grid, how many samples does it take to mine the right number of higher-level structures?' The answer to this question gives us an indication of how aggressive the sampling strategy is, its stability (i.e., once mined, does it continue to mine the patterns?), and comparisons with the other strategy.

5 Experimental Results

We now present empirical results demonstrating the effectiveness of our active mining strategy on both synthetic and real datasets. We employed the Netlab suite of algorithms for GP modeling. Netlab supports a covariance formulation similar to Eq. 3.4, along with a bias term that overcomes our earlier assumption of zero mean. In addition, the model includes a noise term that can capture uncertainties in individual measurements; while this is not required for the deterministic functions considered here, it ensures that the numerical computation doesn't become unstable. All GP parameters are given a relatively broad Gaussian prior. A surrogate model was fit on a regularly spaced grid (more below), with a limit of 100 iterations for conjugate gradient search. The SAL parameters were set to $(1.5, 0.75, 0.1)$, as before. The standard variance-based sampling has no adjustable parameters; a fixed 8-adjacency neighborhood was utilized for defining $P(\mathbf{x})$ in entropy-based sampling. Optimization for Φ_V and Φ_E was conducted over the same grid as the domain of the surrogate function.

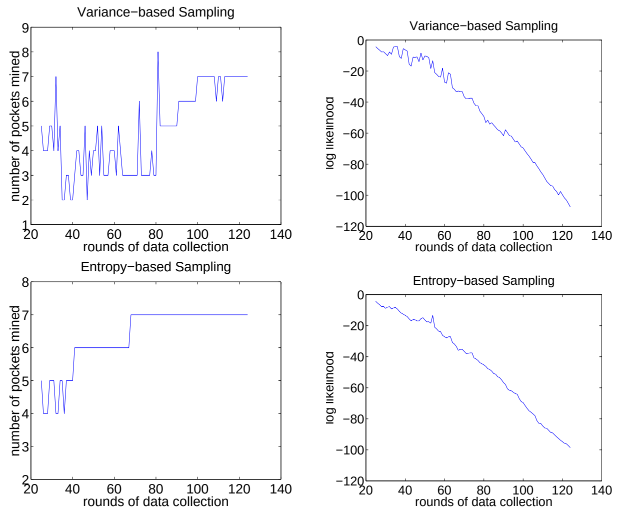


Figure 8: Pocket mining performance on the 7-pocket function from Fig. 7. (top) variance-based and (bottom) entropy-based sampling. (left) number of pockets found and (right) negative log-likelihood.

5.1 Synthetic Datasets: For the synthetic benchmark, we adopted the suite of test functions from [11], an ACM TOMS algorithm to readily generate classes of functions with known local and global minima. The algorithm systematically distorts a convex quadratic function with cubic or quintic polynomials to yield continuously differentiable (D-type) and twice continuously differentiable (D2-type) functions over the closed interval $[-1, 1]^2$. Since our active mining proceeds by discrete search over a pre-defined grid, we evaluated the generated functions over a regular 21×21 grid in $[-1, 1]^2$ ($|G| = 441$) and used these function values as the 'oracle' that is queried by the active mining mechanism. We verified whether in each instance, the SAL miner is able to resolve all pockets when given a complete 21×21 dataset. This is necessary because the radii of the basins of attraction interact with the spacing of the sampling grid, and hence influence the number of samples available for aggregation by the SAL miner. We found that the pocket miner is able to resolve only those generated functions that have up to 7 local minima; functions with more (e.g., 8–12) local minima use only a handful of points (typically 3–9) to represent some of their pockets, too few to be aggregated into a flow class under the SAL miner's parameter settings. Hence, we pruned the automatically generated functions by requiring that that each local minima have at least 12 samples per pocket, when sampled over the 21×21 grid. This yields a collection of 43 functions (21 D-type and 22 D2-type), with numbers of pockets ranging from 4 to 7. Fig. 7 depicts some of these functions.

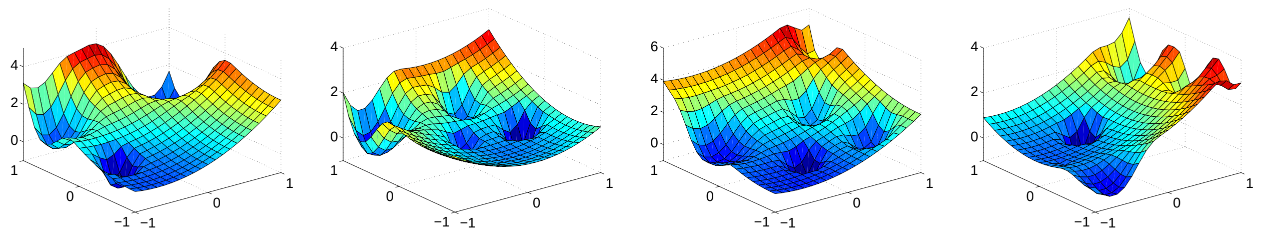


Figure 7: Example test functions with 4, 5, 6, and 7 pockets (respectively). Note that the viewpoint chosen makes visible only some of the pockets in these functions.

Both algorithms were initially seeded with a 5^2 design, comprising 25 points (about 5% of the design space of 441 points). Sampling was conducted for an additional 100 sample values (a total of 125 points, or about 25% of the design space). We reasoned that this is a good interval over which to monitor the performance of the sampling strategies, as even a regularly spaced grid covering 25% of the design space would mine the pockets correctly! Fig. 8 reveals the results for the 7-pocket function of Fig. 7. Both sampling strategies systematically reduce the (negative) log likelihood (as estimated from the GP model parameters) but variance-based sampling shows more oscillatory behavior w.r.t. the number of pockets mined. On close inspection, we found that this strategy goes through stages where adjacent pockets are periodically re-grouped around sample values (which are mostly at the boundaries), causing rapid fluctuations in the SAL miner's output. We say that this strategy is more prone to 'being surprised.' The number of pockets stabilizes around 7 only toward the end of the data collection interval. In contrast, the entropy-based sampling first mines the seven pockets with 68 points, and proceeds to stabilize beyond this point. Similar results have been observed with other test functions.

Next, we analyzed the performance of both algorithms across all 43 test functions. We tested for what fraction of the datasets the mining was correct by, and stayed correct following, a given number of rounds of sampling. Our hypothesis was that the D2-type functions, being smoother, are more easily modeled using GPs and should lend themselves to more aggressive sampling strategies. In addition, the entropy-based sampling strategy should be more effective w.r.t. number of rounds than the variance-based sampling. Fig. 9 shows that this is indeed the case.

5.2 Mining Wireless System Configuration Spaces: Our second application involves characterization of configuration spaces of wireless system designs (see again Fig. 1). The goal is to understand the joint

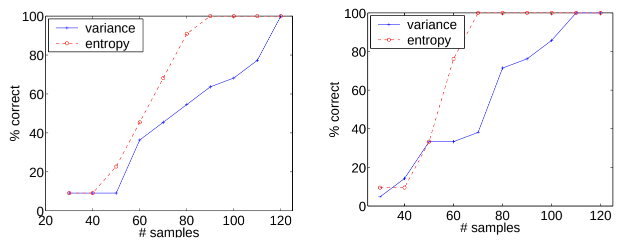


Figure 9: Overall pocket mining performance (fraction of cases correctly identified) with increasing number of samples, for (left) D-type and (right) D2-type functions.

influence of selected configuration parameters on system performance. This can be achieved by identifying spatial aggregates in the configuration space, aggregating low level simulation data (typically multiple samples per configuration point) into regions of constrained shape. In particular, the setup in Fig. 1 is from a study designed to evaluate the performance of STTD (space-time transmit diversity) wireless systems, where the base station uses two transmitter antennas separated by a small distance, in an attempt to improve received signal strength. In this application, the aim is to assess how the power imbalance between the two branches impacts the performance (measured by bit error rate, BER) of the simulated system, across a range of signal-to-noise ratios (SNRs). When the signal components are significant compared to the noise components, and when the SNR ratios of the two branches are comparable, then it is well known that the system would yield high quality of BER performance. What is not so clear is how the performance will degrade as the SNRs move apart. Posed in the spatial aggregation framework, this objective translates into identifying and characterizing (in terms of width, or power imbalance) the pocket in the central portion of the configuration space. Identifying and characterizing other pockets is not as important, since some of them will actually contain suboptimal configurations.

We adopt an experimental methodology similar to that in the previous case studies, and created an 'oracle' from the simulation data described in [26].

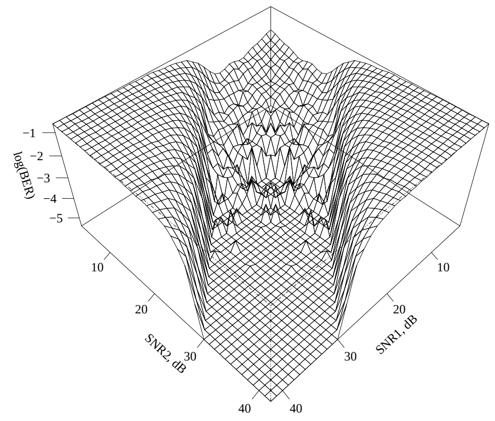


Figure 10: Estimates of BER performance in a space of wireless system configurations.

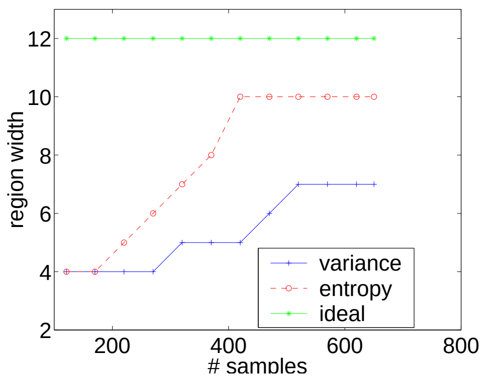


Figure 11: Performance of active mining strategies on wireless simulation data, characterizing width of the main pocket in Fig. 10 with increasing numbers of samples.

Fig. 10 demonstrates that the dataset is quite noisy, especially when the SNR values are low. The design of the oracle, surrogate model building, and sample selection all employ a 55×55 grid over the configuration space (SNR levels ranging from 3dB to 58dB for each antenna). Both variance-based sampling and entropy-based sampling were initialized using a 11^2 design (about 4% of the configuration space). Sampling was conducted for an additional 650 points, yielding a total of 771 points (25% of the design space, as with the earlier studies). For each round of active mining, we determined the majority class occupied by points having equal SNR and determined the maximum width of this class. This measure was periodically tracked across the rounds of data collection. Fig. 11 shows how the sampling strategies fare compared to the correct estimate of 12dB, as reported in [26] by applying a spatial data mining algorithm over the entire dataset. Entropy-based sampling once again selects data that systematically clarify the nature of the pockets, and cause a progressive widening of the trough in the middle. However, it doesn't mine the ideal width of 12dB (within the given samples). We reason that this is because the GP model has difficulty approximating the steep edge of the basin. Variance-based sampling fares worse and demonstrates a slower growth of width across samples. This application highlights the utility of our framework for mining both qualitative and quantitative properties of spatial aggregates.

6 Discussion

This paper has presented a novel integration of approaches from three areas, namely spatial structure discovery, probabilistic modeling using GPs, and active data mining. The spatial aggregation language provides a methodology for identifying multi-level structures in field data, Gaussian processes provide a probabilistic basis for reasoning about uncertainty in field data, and active data mining closes the loop to optimize new samples for uncertainty in field data as well as information content relevant to high-level structures. Entropy-based sampling is suitable whenever we can define information-theoretic functionals over spatial aggregates. In this paper, we have primarily focused on characterizing region boundaries, but this same strategy is applicable to any case where we expect locations with spatial proximity but informational impurity, e.g., identifying breaks and fissures in volumetric data, picking outliers from geographical maps, and detecting violations of coherence in spatio-temporal datasets.

There are several extensions to the work presented here. First, our assumption of sampling over a defined grid can be relaxed and the scope of active mining can be expanded to include subsampling. Second, the modeling of vector fields using GPs warrants further investigation, in particular to address the issue of how to model data fields given only (or also) derivative information or when the underlying function is not smooth or differentiable. Other investigators have done related work in this area [6]. Third, we assume here that the model (of flow classes) posited by SAL is correct, and use this information to drive the sampling. To overcome this assumption, we must create a probabilistic model of SAL's computations (including uncertainty and non-determinism in aggregation procedures) and integrate this model with the GP model for the data fields. Instantiating SAL to popular spatial mining algorithms investigated in the data mining community (e.g. [15, 20]) and applying them in an active mining context is a final direction we are pursuing. These and similar ideas will help establish the many ways in which mathematical models of data approximation can be integrated with data mining algorithms.

Acknowledgements

This work is supported in part by US NSF grants IIS-0237654, EIA-9984317, and IBN-0219332. The wireless simulation dataset is courtesy of Alex Verstak.

References

[1] C. Bailey-Kellogg and N. Ramakrishnan. Ambiguity-Directed Sampling for Qualitative Analysis of Sparse Data from Spatially Distributed Physical Systems. In *Proc. IJCAI*, pages 43–50, 2001.

[2] C. Bailey-Kellogg and F. Zhao. Influence-Based Model Decomposition for Reasoning about Spatially Distributed Physical Systems. *Artificial Intelligence*, Vol. 130(2):pages 125–166, 2001.

[3] C. Bailey-Kellogg, F. Zhao, and K. Yip. Spatial Aggregation: Language and Applications. In *Proc. AAAI*, pages 517–522, 1996.

[4] K. Brinker. Incorporating Diversity in Active Learning with Support Vector Machines. In *Proceedings of the Twentieth International Conference on Machine Learning (ICML'03)*, pages 59–66, 2003.

[5] D.A. Cohn, Z. Ghahramani, and M.I. Jordan. Active Learning with Statistical Models. *Journal of Artificial Intelligence Research*, Vol. 4:pages 129–145, 1996.

[6] D. Cornford, I.T. Nabney, and C.K.I. Williams. Adding Constrained Discontinuities to Gaussian Process Models of Wind Fields. In *Proceedings of NIPS*, pages 861–867, 1998.

[7] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge University Press, 2000.

[8] C. Currin, T. Mitchell, M. Morris, and D. Ylvisaker. Bayesian Prediction of Deterministic Functions, with Applications to the Design and Analysis of Computer Experiments. *J. Amer. Stat. Assoc.*, Vol. 86:pages 953–963, 1991.

[9] R.G. Easterling. Comment on 'Design and Analysis of Computer Experiments'. *Statistical Science*, 4(4):425–427, 1989.

[10] J. Garcke and M. Griebel. Data Mining with Sparse Grids using Simplicial Basis Functions. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 87–96, 2001.

[11] M. Gaviano, D.E. Kvasov, D. Lera, and Y.D. Sergeyev. Algorithm 829: Software for Generation of Classes of Test Functions with Known Local and Global Minima for Global Optimization. *ACM Transactions on Mathematical Software*, Vol. 29(4):pages 469–480, Dec 2003.

[12] X. Huang and F. Zhao. Relation-Based Aggregation: Finding Objects in Large Spatial Datasets. In *Proceedings of the 3rd International Symposium on Intelligent Data Analysis*, 1999.

[13] J.-N. Hwang, J.J. Choi, S. Oh, and R.J. Marks II. Query-based Learning Applied to Partially Trained Multilayer Perceptrons. *IEEE Transactions on Neural Networks*, Vol. 2(1):pages 131–136, 1991.

[14] A.G. Journel and C.J. Huijbregts. *Mining Geostatistics*. Academic Press, New York, 1992.

[15] G. Karypis, E.-H. Han, and V. Kumar. Chameleon: Hierarchical Clustering using Dynamic Modeling. *IEEE Computer*, Vol. 32(8):pages 68–75, 1999.

[16] J. Koehler and A. Owen. Computer Experiments. In S. Ghosh and C. Rao, editors, *Handbook of Statistics: Design and Analysis of Experiments*, pages 261–308. North Holland, 1996.

[17] D.J. MacKay. Information-Based Objective Functions for Active Data Selection. *Neural Computation*, Vol. 4(4):pages 590–604, 1992.

[18] I.T. Nabney. *Netlab: Algorithms for Pattern Recognition*. Springer-Verlag, 2002.

[19] R.M. Neal. Monte Carlo Implementations of Gaussian Process Models for Bayesian Regression and Classification. Technical Report 9702, Department of Statistics, University of Toronto, Jan 1997.

[20] R.T. Ng and J. Han. CLARANS: A Method for Clustering Objects for Spatial Data Mining. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 14(5):pages 1003–1016, 2002.

[21] I. Ordóñez and F. Zhao. STA: Spatio-Temporal Aggregation with Applications to Analysis of Diffusion-Reaction Phenomena. In *Proc. AAAI*, pages 517–523, 2000.

[22] N. Ramakrishnan and C. Bailey-Kellogg. Sampling Strategies for Mining in Data-Scarce Domains. *IEEE/AIP CiSE*, Vol. 4(4):pages 31–43, 2002.

[23] N. Ramakrishnan and C. Bailey-Kellogg. Gaussian Process Models of Spatial Aggregation Algorithms. In *Proc. IJCAI*, pages 1045–1051, 2003.

[24] J. Sacks, W.J. Welch, T.J. Mitchell, and H.P. Wynn. Design and Analysis of Computer Experiments. *Statistical Science*, Vol. 4(4):pages 409–435, 1989.

[25] S. Tong and D. Koller. Support Vector Machine Active Learning with Applications to Text Classification. *Journal of Machine Learning Research*, Vol. 2:pages 45–66, 2001.

[26] A. Verstak et al. Using Hierarchical Data Mining to Characterize Performance of Wireless System Configurations. Technical Report cs.CE/0208040, CoRR, Aug 2002.

[27] C.K.I. Williams. Prediction with Gaussian Processes: From Linear Regression to Linear Prediction and Beyond. In M.I. Jordan, editor, *Learning in Graphical Models*, pages 599–621. MIT Press, Cambridge, MA, 1998.

[28] K.M. Yip and F. Zhao. Spatial Aggregation: Theory and Applications. *JAIR*, Vol. 5:pages 1–26, 1996.

[29] K.M. Yip, F. Zhao, and E. Sacks. Imagistic Reasoning. *ACM Computing Surveys*, Vol. 27(3):pages 363–365, 1995.

[30] F. Zhao, C. Bailey-Kellogg, and M.P.J. Fromherz. Physics-Based Encapsulation in Embedded Software for Distributed Sensing and Control Applications. *Proceedings of the IEEE*, 91:40–63, 2003.