Model-based Querying in Sensor Networks

Amol Deshpande, University of Maryland, http://www.cs.umd.edu/~amol

Carlos Guestrin, Carnegie Mellon University, http://www.cs.cmu.edu/~guestrin

Samuel Madden, MIT, http://db.lcs.mit.edu/madden

SYNONYMS

Approximate querying; Model-driven data acquisition

DEFINITION

The data generated by sensor networks or other distributed measurement infrastructures is typically incomplete, imprecise, and often erroneous, such that it is not an accurate representation of physical reality. To map raw sensor readings onto physical reality, a mathematical description, a *model*, of the underlying system or process is required to complement the sensor data. Models can help provide more robust interpretations of sensor readings: by accounting for spatial or temporal biases in the observed data, by identifying sensors that are providing faulty data, by extrapolating the values of missing sensor data, or by inferring hidden variables that may not be directly observable. Models also offer a principled approach to predict future states of a system. Finally, since models incorporate spatio-temporal correlations in the environment (which tend to be very strong in many monitoring applications), they lead to significantly more energy-efficient query execution – by exploiting such attribute correlations, it is often possible to use a small set of observations to provide approximations of the values of a large number of attributes.

Model-based querying over a sensor network consists of two components: (1) identifying and/or building a model for a given sensor network, and (2) executing declarative queries against a sensor network that has been augmented with such a model (these steps may happen serially or concurrently). The queries may be on future or hidden states of the system, and are posed in a declarative SQL-like language. Since the cost of acquiring sensor readings from the sensor nodes is the dominant cost in these scenarios, the optimization goal typically is to minimize the total data acquisition cost.

HISTORICAL BACKGROUND

Statistical and probabilistic models have been a mainstay in the scientific and engineering communities for a long time, and are commonly used for a variety of reasons, from simple pre-processing tasks to remove noise (e.g., using Kalman Filters) to complex analysis tasks for prediction purposes (e.g., to predict weather or traffic flow). Standard books on machine learning and statistics should be consulted for more details (e.g., Cowell [3], Russell and Norvig [14]).

The first work to combine models, declarative SQL-like queries and live data acquisition in sensor networks was the BBQ System [7, 6]. The authors proposed a general architecture for model-based querying, and posed the optimization problem of selecting the best sensor readings to acquire to satisfy a user query (which can be seen as a generalization of the *value of information problem* [14]). The authors proposed several algorithms for solving this optimization problem; they also evaluated the approach on a several real-world sensor network datasets, and demonstrated that model-based querying can provide high-fidelity representation of the real phenomena and leads to significant performance gains versus traditional data acquisition techniques. Several works since then have considerably expanded upon the basic idea, including development of sophisticated algorithms for data acquisition [12, 13], more complex query types [15], and integration into a relational database system [8, 11].

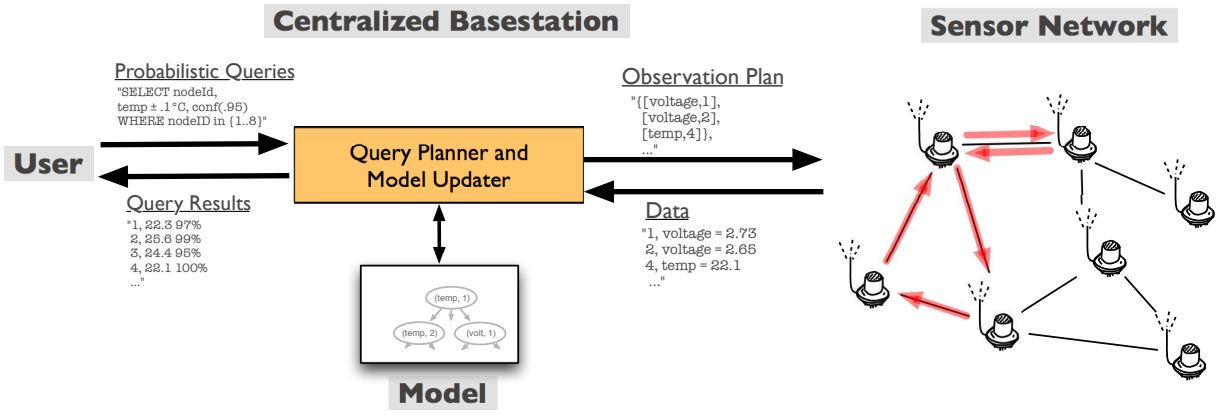


Figure 1: Architecture of a Model-based Querying System (adapted from BBQ [7, 6])

The querying aspect of this problem has many similarities to the problem of approximate query processing in database systems, which often uses model-like *synopses*. For example, the AQUA project [1] proposed a number of sampling-based synopses that can provide approximate answers to a variety of queries using a fraction of the total data in a database. As with BBQ, such answers typically include tight bounds on the correctness of answers. AQUA, however, is designed to work in an environment where it is possible to generate an independent random sample of data (something that is quite tricky to do in sensor networks, as losses are correlated and communicating random samples may require the participation of a large part of the network). AQUA also does not exploit correlations, which means that it lacks the *predictive* power of representations based on probabilistic models. Deshpande et al. [4] and Getoor et al. [9] proposed exploiting data correlations through use of graphical modeling techniques for approximate query processing, but, unlike BBQ, neither provide any guarantees on the answers returned. Furthermore, the optimization goal of approximate query processing is typically not to minimize the data acquisition cost, rather it is minimizing the size of the synopsis, while maintaining reasonable accuracy.

SCIENTIFIC FUNDAMENTALS

Figure 1 shows the most common architecture of a model-based querying system (adapted from the architecture of the BBQ system). The model itself is located at a centralized, Internet-connected basestation, which also interacts with the user. The user may issue either continuous or ad-hoc queries against the sensor network, using a declarative SQL-like language. The key module in this architecture is the *query planner and model updater*, which is in charge of maintaining the model and answering the user queries (possibly by acquiring more data from the underlying sensor network). The following sections elaborate on the various components of such a system.

Model

A model is essentially a simplified representation of the underlying system or the process, and describes how various attributes of the system interact with each other, and how they evolve over time. Hence, the exact form of the model is heavily dependent on the system being modeled, and an astounding range of models have been developed over the years for different environments. For ease of exposition, the rest of this entry focuses on a dynamic model similar to the one used in the BBQ system.

Let $X_1, \cdots, X_n$ denote the (n) attributes of interest in the sensor network. Further, let X_i^t denote the value of X_i at time t (assuming that time is discrete). At any time t, a subset of these attributes may be observed and communicated to the basestation; here, the observations at time t are denoted by $\mathbf{o}^t$ (note that hidden variables can never be observed). The attributes typically correspond to the properties being monitored by the sensor nodes (e.g., *temperature* on sensor number 5, *voltage* on sensor number 8). However, more generally, they may be *hidden variables* that are of interest, but cannot be directly observed. For example, it may be useful to model and query a hidden boolean variable that denotes whether a sensor is faulty [11] – the value of this variable can

be inferred using the model and the actual observations from the sensor.

Loosely speaking, the model encodes the spatial and temporal relationships between these attributes of interest. At any time t, the model provides us with a posterior *probability density function* (pdf), $p(X_1^t, \cdots, X_n^t | \mathbf{o}^{1\cdots(t-1)})$, assigning a probability for each possible assignment to the attributes at time t given the observations made so far. Such a joint distribution can capture all the spatial correlations between the attributes; more compact representations like Bayesian networks can be used instead as well.

To model the temporal correlations, it is common to make a *Markov* assumption; given the values of *all* attributes at time t, one assumes that the values of the attributes at time $t+1$ are independent of those for any time earlier than t. This assumption leads to a simple model for a dynamic system where the dynamics are summarized by a conditional density called the *transition model*, $p(X_1^{t+1}, \ldots, X_n^{t+1} \mid X_1^t, \ldots, X_n^t)$. Using a transition model, one can compute $p(X_1^{t+1}, \ldots, X_n^{t+1} \mid \mathbf{o}^{1\cdots t})$ from $p(X_1^t, \ldots, X_n^t \mid \mathbf{o}^{1\cdots t})$ using standard probabilistic procedures. Different transition models may be used for different time periods (e.g., hour of day, day of week, season, etc.) to model the differences in the way the attributes evolve at different times.

Learning the model: Typically in probabilistic modeling, a *class* of models is chosen (usually with input from a *domain expert*), and learning techniques are then used to pick the best model in the class. Model parameters are typically learned from training data, but can also be directly inferred if the behavior of the underlying physical process is well-understood. In BBQ, the model was learned from historical data, which consisted of readings from all of the monitored attributes over some period of time.

Updating the model: Given the formulation above, model updates are fairly straightforward. When a new set of observations arrives (say $\mathbf{o}^t$), it can be incorporated into the model by conditioning on the observations to compute new distributions (ie., by computing $p(X_1^t, \ldots, X_n^t \mid \mathbf{o}^{1\cdots t})$ from $p(X_1^t, \ldots, X_n^t \mid \mathbf{o}^{1\cdots(t-1)})$. Similarly, as time advances, the transition model is used to compute the new distribution for time $t+1$ (ie., $p(X_1^{t+1}, \ldots, X_n^{t+1} \mid \mathbf{o}^{1\cdots t})$ is computed from $p(X_1^t, \ldots, X_n^t \mid \mathbf{o}^{1\cdots t})$ and $p(X_1^{t+1}, \ldots, X_n^{t+1} \mid X_1^t, \ldots, X_n^t)$).

Query Planning and Execution

User queries are typically posed in a declarative SQL-like language that may be augmented with constructs that allow users to specify the approximation that the user is willing to tolerate, and the desired confidence in the answer. For example, the user may ask the system to report the temperature readings at all sensors within ± 0.5, with confidence 95%. For many applications (e.g., building temperature control), such approximate answers may be more than sufficient. Tolerance for such approximations along with the correlations encoded by the model can lead to significant energy savings in answering such queries.

Answering queries probabilistically based on a pdf over the query attributes is conceptually straightforward; to illustrate this process, consider two types of queries (here, assume that all queries are posed over attributes $X_1^t, \cdots, X_n^t$, and the corresponding pdf is given by $p(X_1^t, \cdots, X_n^t)$):

Value query: A value query [6] computes an approximation of the values of the attributes to within $\pm\epsilon$ of the true value, with confidence at least $1 - \delta$. Answering such a query involves computing the expected value of each of the attribute, μ_i^t, using standard probability theory. These μ_i^t's will be the reported values. The pdf can then be used again to compute the probability that X_i^t is within ϵ from the mean, $P(X_i^t \in [\mu_i^t - \epsilon, \mu_i^t + \epsilon])$. If all of these probabilities meet or exceed user specified confidence threshold, then the requested readings can be directly reported as the means μ_i^t. If the model's confidence is too low, additional readings must be acquired before answering the query (see below).

Max query: Consider an *entity* version of this query [2] where the user wants to know the identity of the sensor reporting the maximum value. A naive approach to answering this query is to compute, for each sensor, the probability that its value is the maximum. If the maximum of these probabilities is above $1 - \delta$, then an answer can be returned immediately; otherwise, more readings must be acquired. Although conceptually simple, computing the probability that a given sensor is reporting the maximum value is non-trivial, and requires complex integration that may be computationally infeasible [15].

If the model is not able to provide sufficient confidence in the answer, the system must acquire more readings from the sensor network, to bring the model's confidence up to the user specified threshold. Suppose the system observes a set of attributes $\mathcal{O} \subset \{X_1, ..., X_n\}$. After incorporating these observations into the model and recomputing the answer, typically the confidence in the (new) answer will be higher (this is not always true). The new confidence will typically depend on the actual observed values. Let $R(\mathcal{O})$ denote the *expected* confidence in the answer after observing $\mathcal{O}$. Then, the optimization problem of deciding which attributes to observe can be stated as follows:

$$\begin{array}{ll} \text{minimize}_{\mathcal{O} \subseteq \{1,...,n\}} & C(\mathcal{O}), \\ \text{such that} & R(\mathcal{O}) \geq 1 - \delta. \end{array} \quad (1)$$

where $C(\mathcal{O})$ denotes the *data acquisition cost* of observing the values of attributes in $\mathcal{O}$.

This optimization problem combines three problems that are known to be intractable, making it very hard to solve it in general:

- Answering queries using a pdf: as mentioned above, this can involve complex numerical integration even for simple queries such as max.
- Choosing the minimum set of sensor readings to acquire to satisfy the query (this is similar to the classic *value of information problem*) [14, 12, 15, 10].
- Finding the optimal way to collect a required set of sensor readings from the sensor network that minimizes the total communication cost. Meliou et al. [13] present several approximation algorithms for this NP-Hard problem.

Example

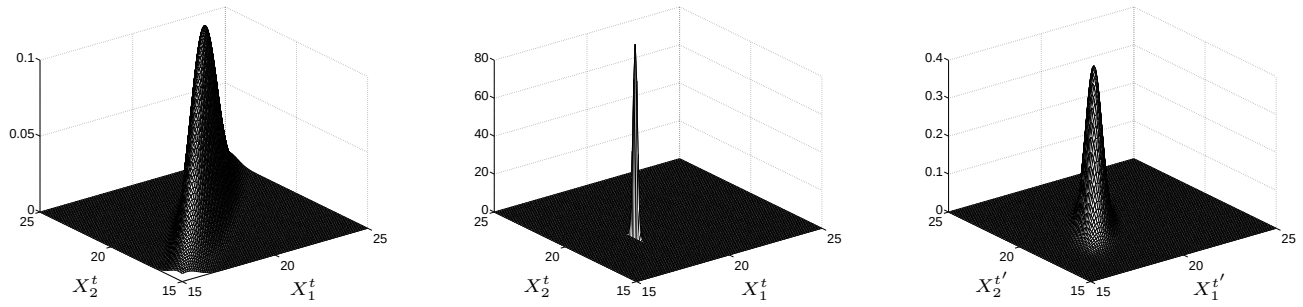
Figure 2 illustrates the query answering process using a simple example, where the model takes the form of time-varying *bivariate Gaussian (normal)* distribution over two attributes, X_1 and X_2. This was the basic model used in the BBQ system. A bivariate Gaussian is the natural extension of the familiar unidimensional normal probability density function (pdf), known as the "bell curve". Just as with its 1-dimensional counterpart, a bivariate Gaussian can be expressed as a function of two parameters: a length-2 vector of means, μ, and a 2×2 matrix of covariances, Σ. Figure 2 (i) shows a three-dimensional rendering of a Gaussian over the two attributes at time t, X_1^t and X_2^t; the z axis represents the *joint density* that $X_2^t = x$ and $X_1^t = y$.

Now, consider a *value query* over this model, posed at time t, which asks for the values of X_1^t and X_2^t, within $\pm\epsilon$, with confidence $1 - \delta$. The reported values in this case would be the means (μ), and the confidence can be computed easily using Σ (details can be found in [6]). Considering the high initial covariance, it is unlikely that the Gaussian in Figure 2 (i) can achieve the required confidence. Suppose the system decides to observe X_1^t. Figure 2 (ii) shows the result of incorporating this observation into the model. Note that not only does the spread of X_1^t reduces to near zero, because of the high correlation between X_1^t and X_2^t, the variance of X_2^t also reduces dramatically, allowing the system to answer the query with required confidence.

Then, after some time has passed, the belief about the values of X_1 and X_2 (at time $t' > t$) will be "spread out", again providing a high-variance Gaussian over two attributes, although both the mean and variance may have shifted from their initial values, as shown in Figure 2 (iii).

KEY APPLICATIONS

Model-based querying systems like BBQ, that exploit statistical modeling techniques and optimize the utilization of a network of resource constrained devices could have significant impact in a number of application domains, ranging from control and automation in buildings [16] to highway traffic monitoring. Integrating a model into the data acquisition process can significantly improve data quality and reduce data uncertainty. The ability to query over missing, future or hidden states of the system will prove essential in many applications where sensor failures are common (e.g., highway traffic monitoring) or where direct observation of the variables of interest is not feasible. Finally, model-based querying has the potential to significantly reduce the cost of data acquisition, and thus can improve the life of a resource-constrained measurement infrastructure (e.g., battery-powered wireless



(i) (ii) (iii)

Figure 2: Example of Gaussians: (i) 3D plot of a 2D Gaussian with high covariance; (ii) the resulting Gaussian after a particular value of X_1^t has been observed (because of measurement noise, there might still be some uncertainty about the true value of X_1^t); (iii) the uncertainty about X_1 and X_2 increases as time advances to t'.

sensor networks) manyfold.

FUTURE DIRECTIONS

Model-based querying is a new and exciting research area with many open challenges that are bound to become more important with the increasingly widespread use of models for managing sensor data. The two most important challenges are dealing with a wide variety of models that may be used in practice, and designing algorithms for query processing and data acquisition; these are discussed briefly below:

Model selection and training: The choice of model affects many aspects of model-based querying, most importantly the accuracy of the answers and the confidence bounds that can be provided with them. The problem of selecting the right model class has been widely studied [14, 3] but can be difficult in some applications. Furthermore, developing a new system for each different model is not feasible. Ideally, using a new model should involve little to no effort on the part of user. Given a large variety of models that may be applicable in various different scenarios, this may turn out to be a tremendous challenge.

Algorithms for query answering and data acquisition: Irrespective of the model selected, when and how to acquire data in response to a user query raises many hard research challenges. As discussed above, this problem combines three very hard problems, and designing general-purpose algorithms that can work across the spectrum of different possible model remains an open problem.

See Deshpande et al. [5] for a more elaborate discussion of the challenges in model-based querying.

CROSS REFERENCE

Data uncertainty management in sensor networks; Ad-hoc queries in sensor networks; Approximate query processing; Temporal probabilistic models.

RECOMMENDED READING

[1] Swarup Acharya, Phillip B. Gibbons, Viswanath Poosala, and Sridhar Ramaswamy. Join synopses for approximate query answering. In *SIGMOD '99: Proceedings of the 1999 ACM SIGMOD international conference on Management of data*, pages 275–286, New York, NY, USA, 1999. ACM Press.

[2] Reynold Cheng, Dmitri V. Kalashnikov, and Sunil Prabhakar. Evaluating probabilistic queries over imprecise data. In *SIGMOD '03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 551–562, New York, NY, USA, 2003. ACM Press.

[3] R. Cowell, P. Dawid, S. Lauritzen, and D. Spiegelhalter. *Probabilistic Networks and Expert Systems.* Spinger, New York, 1999.

[4] Amol Deshpande, Minos Garofalakis, and Rajeev Rastogi. Independence is good: dependency-based histogram

synopses for high-dimensional data. In *SIGMOD '01: Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, pages 199–210, New York, NY, USA, 2001. ACM Press.

[5] Amol Deshpande, Carlos Guestrin, and Sam Madden. Using probabilistic models for data management in acquisitional environments. In *CIDR '05: Proceedings of the Second Biennial Conference on Innovative Data Systems Research*, 2005.

[6] Amol Deshpande, Carlos Guestrin, Sam Madden, Joe Hellerstein, and Wei Hong. Model-driven approximate querying in sensor networks. *The VLDB Journal*, 14(4):417–443, 2005.

[7] Amol Deshpande, Carlos Guestrin, Samuel Madden, Joseph M. Hellerstein, and Wei Hong. Model-driven data acquisition in sensor networks. In *VLDB '04: Proceedings of the Thirtieth International Conference on Very Large Data Bases, Toronto, Canada*, 2004.

[8] Amol Deshpande and Samuel Madden. MauveDB: supporting model-based user views in database systems. In *SIGMOD '06: Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 73–84, New York, NY, USA, 2006. ACM Press.

[9] Lise Getoor, Benjamin Taskar, and Daphne Koller. Selectivity estimation using probabilistic models. In *SIGMOD '01: Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, pages 461–472, New York, NY, USA, 2001. ACM Press.

[10] Ashish Goel, Sudipto Guha, and Kamesh Munagala. Asking the right questions: model-driven optimization using probes. In *PODS '06: Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 203–212, New York, NY, USA, 2006. ACM Press.

[11] B. Kanagal and A. Deshpande. Online filtering, smoothing and probabilistic modeling of streaming data. In *ICDE 2008*, 2008.

[12] A. Krause, C. Guestrin, A. Gupta, and J. Kleinberg. Near-optimal sensor placements: maximizing information while minimizing communication cost. *IPSN '06: Proceedings of the fifth international conference on Information processing in sensor networks*, pages 2–10, 2006.

[13] A. Meliou, D. Chu, J. Hellerstein, C. Guestrin, and W. Hong. Data gathering tours in sensor networks. *IPSN '06: Proceedings of the fifth international conference on Information processing in sensor networks*, pages 43–50, 2006.

[14] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 1994.

[15] Adam Silberstein, Rebecca Braynard, Carla Ellis, Kamesh Munagala, and Jun Yang. A sampling-based approach to optimizing top-k queries in sensor networks. In *ICDE '06: Proceedings of the 22nd International Conference on Data Engineering (ICDE'06)*, page 68, Washington, DC, USA, 2006. IEEE Computer Society.

[16] V. Singhvi, A. Krause, C. Guestrin, J.H. Garrett Jr, and H.S. Matthews. Intelligent light control using sensor networks. *Sensys '05: Proceedings of the 3rd international conference on Embedded networked sensor systems*, pages 218–229, 2005.