Machine Learning in Ecosystem Informatics and Sustainability

Thomas G. Dietterich
School of Electrical Engineering and Computer Science
Oregon State University
tgd@cs.orst.edu

Abstract

Ecosystem Informatics brings together mathematical and computational tools to address scientific and policy challenges in the ecosystem sciences. These challenges include novel sensors for collecting data, algorithms for automated data cleaning, learning methods for building statistical models from data and for fitting mechanistic models to data, and algorithms for designing optimal policies for biosphere management. This presentation discusses these challenges and then describes recent work on the first two of these—new methods for automated arthropod population counting and linear Gaussian DBNs for automated cleaning of sensor network data.

1 Introduction

Computer science has had a revolutionary impact in molecular biology and genetics. This impact was not merely the result of automating existing ways of doing science. Instead, novel computer science methods, when coupled with novel instruments (e.g., shotgun sequencing of the genome, DNA arrays) transformed the scientific enterprise. In place of hypothesis-driven experiments examining a particular subsystem or pathway, computational methods supported data-driven science in which massive amounts of data were collected first, and then subjected to computational analysis to suggest hypotheses and fit statistical and causal models.

This change was initially controversial. Scientists and funding agencies were concerned that data collected without any prior hypothesis would be useless. But whole genome sequencing has turned out to be hugely important for addressing a wide range of questions in molecular and cell biology as well as evolution and population biology.

The ecosystem sciences (ecosystem ecology, community ecology, landscape ecology, hydrology, etc.) are today where molecular biology was in the mid-1990s. Most research projects formulate hypotheses and perform manipulative experiments to refine and test them. Consequently, progress is slow, and many of the most important management questions (e.g., preventing species extinctions, limiting the spread of invasive species and diseases, restoring ecosystems to healthy function, mitigating the effects of climate change) cannot be answered by the current state of scientific knowledge. There is broad agreement that the ecosystem sciences are data-limited, and there are several efforts under way to collect observational data on a much larger scale than in the past (e.g., the National Ecological Observatory Network (NEON; `www.neoninc.org`). As ecology becomes a data-driven science, there is a great need for computer scientists to help with the entire data pipeline from instruments, to data management, to model fitting, to policy making. Figure 1 shows the data pipeline. Sensors capture data to create datasets. These are then analyzed to produced models that can support the design of policies. Models also guide the formation of hypotheses which can then be tested by designing and executing experiments. There are many opportunities to apply advanced computer science and artificial intelligence methods in this pipeline.

- **Sensor Algorithms**. Many sensors incorporate complex algorithms to transform the raw signals into meaningful data. For example, in Section 2 below, I will describe the application of computer vision methods to classify and count arthropod specimens.
- **Data Cleaning**. Sensors fail, particularly when they are placed in challenging environments (glaciers, mountain tops, the seafloor). When data is collected at large scale, it is no longer feasible for people to manually detect and diagnose sensor failures. Automated data cleaning methods are needed that can detect and correct sensor failures in real time.
- **Model Fitting**. Once datasets are constructed, models can be fit to them. The two primary kinds of models—predictive models and causal models—are both needed for ecological science and ecosystem management. A challenging aspect of ecological models is that many different kinds of data, at many different spatial and temporal scales, need to be considered simultaneously. An example of predictive models are species distribution models [Elith *et al.*, 2006]. These attempt to predict the spatio-temporal distribution of plant and animal species as a function of climate and

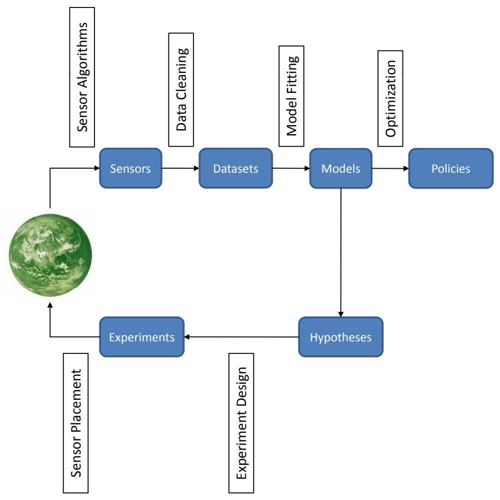


Figure 1: The Ecosystem Informatics Pipeline

habitat. A particular challenge is to jointly predict the distribution of thousands of plant and animal species in order to guide the design of conservation policies.

- **Optimization**. The development of optimal policies for ecosystem management usually involves solving optimization problems constrained by causal or predictive models. In ecological settings, the objective function typically incorporates both biological goals (e.g., species survival, biodiversity) and economic goals (e.g., ecosystem services such as water filtration, pollination, timber, agriculture). An important issue is to take into account the uncertainty resulting from the previous steps in the pipeline.
- **Experiment Design**. Although the primary (upper) pipeline focuses on observational data, there is still a need for hypothesis-driven data collection. Methods for experiment design can help scientists construct experiments that are inexpensive and maximize the scientific value of the information that is collected.
- **Sensor Placement**. A special case of experiment design is to determine where to place sensors (and possibly, where to move them) to maximize their scientific effectiveness [Krause *et al.*, 2008].

Over the past decade at Oregon State University, we have launched research and education efforts to address these computational challenges. In the remainder of this presentation, I will discuss two of our current activities: (a) rapid throughput arthropod identification and (b) automated data cleaning for sensor networks. This talk is an invitation to join us in addressing these important and challenging problems.

2 Rapid Throughput Arthropod Identification

The arthropods (insects, arachnids, crustaceans, millipedes, and centipedes) form a huge branch of the tree of life. Arthropods are found in virtually all environments on earth including lakes, streams, soils, oceans, and on animals. In typical food webs, arthropods consume the primary producers (bacteria and plants) and in turn are consumed by "higher" animals such as birds and mammals. Consequently, arthropods are very good indicators of ecosystem functioning. They provide convenient measures of biodiversity and ecosystem health, so they are important dependent variables for understanding and restoring ecosystems.

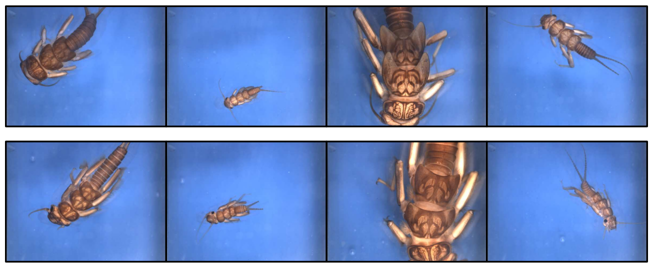


Figure 2: Sample images from our STONEFLY9 dataset. The top and bottom rows show two distinct stonefly species, difficult to classify even for trained human experts.

Arthropods are very easy to collect. Unfortunately, while some arthropods are easy to distinguish, many species are only subtly different from one another. Hence, the effective exploitation of arthropod data is limited by the number of (expensive) experts available to manually classify and count the specimens.

To address this problem, we launched the BugID project whose goal is to develop robotic devices and associated computer vision algorithms for automatically manipulating, photographing, classifying, and separating arthropod specimens. The first problem that we focused on was identifying stonefly larvae. Stoneflies live in the substrate of freshwater streams, and they are sensitive indicators of pollution. Figure 2 shows example images for two stonefly species. These were collected using a robotic apparatus that manipulates the specimens into the field of view of a microscope via pumps and alcohol jets and captures images via a computer-controlled camera [Larios *et al.*, 2008].

We formulated the problem of classifying these insect images as a problem of generic object recognition. Much recent research in computer vision and machine learning has studied this problem. The state-of-the-art approach is based on the following sequence of steps:

- **Detection.** Apply one or more interest detectors to the image to find interest regions. Detectors include the Harris and Hessian detectors [Mikolajczyk and Schmid, 2002], our own PCBR detector [Deng *et al.*, 2007], and the Kadir-Brady salient region detector [Kadir and Brady, 2001]. These are regions of the image of various sizes.
- **Description.** Describe each interest region by a descriptor vector. The 128-dimensional SIFT descriptor

[Lowe, 2004] is by far the most popular. Each image i is now represented by a bag $B_i = \{x_{i,1}, \ldots, x_{i,N_i}\}$ of descriptor vectors. The key machine learning challenge is to develop a classifier architecture that can handle these bags of descriptors.

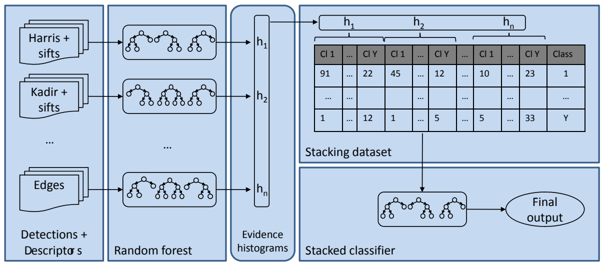
- **Dictionary Learning.** The dominant architecture is to take all of the descriptor vectors in a training set of images and cluster them (e.g., via k-means clustering) into D clusters. In our work, we form a separate dictionary for each class (e.g., species). Let $w_{j,k}$ be cluster j for class k. These are often referred to as keywords.
- **Conversion to Histograms.** Convert the bag B_i into a histogram H_i such that $H_{i,j,k}$ is the number of elements in B_i that were assigned to cluster $w_{j,k}$. Note that each descriptor vector $x_{i,l}$ is mapped to one cluster in *each* of the D dictionaries.
- **Classifier Training and Testing.** Each training example is now represented by a fixed length histogram feature vector H_i (containing $D \times K$ elements, where D is the number of clusters in each dictionary and K is the number of classes). Hence, any standard machine learning classification method can be applied.

In practice, the available data is split into three parts: clustering, training, and testing. The clustering data is used only for learning the dictionary. The training data is then re-represented as histograms using the learned dictionary. And the testing data is employed to evaluate performance.

Our entomology collaborators collected specimens for 9 taxa of stoneflies from streams in Oregon. These were photographed using our apparatus, and good dorsal views were selected. The result is our STONEFLY9 dataset, which is available for download from `web.engr.oregonstate.edu/~tgd/bugid/`.

We applied the standard dictionary approach outlined above to STONEFLY9. However, the results were mediocre: 16.1% error. We hypothesize that there are at least two reasons for this poor performance. First, the dictionaries are learned using purely unsupervised methods—they do not take into account the performance task. Second, information is lost when each descriptor vector is mapped to a dictionary entry. Mapping a SIFT vector to a 2700-word dictionary retains at most 12 bits of information, whereas the original SIFT vector contains 1024 bits. To address the first problem, several researchers including ourselves have developed quasi-supervised methods for creating visual dictionaries [Moosmann *et al.*, 2007; Winn *et al.*, 2005; Yang *et al.*, 2008; Zhang and Dietterich, 2008] However, we have recently developed two methods that both yield very big increases in performance by using simpler methods. We now describe these two methods.

The first method is called *stacked random forests* [Martínez-Muñoz *et al.*, 2009]. Given a set of training bags of the form (B_i, y_i), where B_i is the bag of SIFT vectors and y_i is the class label, we generate labeled instances by pushing the training labels down to the individual descriptor vectors: $(x_{i,l}, y_i)$. Next, for each type of detector (Harris, Hessian, PCBR, Kadir-Brady), we train a random forest [Breiman, 2001] to predict the class of the image from individual detections. We modify the random forest algorithm in two ways. First, we constrain each decision tree so that every leaf contains at least 20 training examples. Second, we store in each leaf ℓ a histogram h_ℓ whose kth entry is the number of training examples from class k that reached this leaf. We call h_ℓ the *evidence histogram* for leaf ℓ.



Cl 1	...	Cl Y	Cl 1	...	Cl Y	...	Cl 1	...	Cl Y	Class
91	...	22	45	...	12	...	10	...	23	1
...	...			...		...		...		...
1	...	12	1	...	5	...	5	...	33	Y

Figure 3: Stacked Random Forest Architecture

To classify a new image (represented as a bag B of descriptor vectors), we take each descriptor vector and "drop" it through all of these random forests. We then take the vector sum of the leaf evidence histograms to create a K-element histogram, which is then normalized to form a probability distribution. A separate K-element histogram is obtained from each random forest. These are then concatenated and fed to the second-level "stacked" classifier to make the final decision.

To train this stacked classifier, we use the "out-of-bag" strategy. Recall that when tree τ in the random forest is constructed, it is learned from a bootstrap sample of the original training data. This means that for each tree τ, there is a set of training images that were not used to build that tree (aka "out of bag"). To construct a stacking example for image i, we process the descriptor vectors in B_i through each tree τ for which image i was out-of-bag. We form the evidence histograms, normalize them, and concatenate them. After the stacking examples are formed, we apply C4.5 with 200 iterations of Adaboost [Freund and Schapire, 1997] to learn the stacking classifier. This process is summarized in Figure 3.

The second method that we developed is even simpler. The basic idea is just to apply boosting to the classic dictionary method [Zhang *et al.*, 2009]. Specifically, we do the following. We employ 3 detectors (Hessian, Kadir-Brady, and PCBR). The outer loop consists of 30 iterations of Adaboost. We employ boosting by sampling, so after updating the weights of the images via the standard Adaboost method, we sample only 20% of the data set (with replacement) using a slight modification of weighted sampling known as Quasi-Random Weighted Sampling [Kalal *et al.*, 2008]. Once a sample of images is drawn, the descriptor vectors in the image are used to learn one $D = 100$ cluster dictio-

Table 1: Error rates of 3 classification architectures on the STONEFLY9 data set

Method	Error rate
Class-specific dictionaries + Adaboost	16.1
Stacked random forests	6.4
Boosted dictionaries	4.9

nary for each detector via k-means clustering. Unlike the class-specific dictionaries described above, these dictionaries are learned from all detections of all classes. After learning the dictionary, we map each training image into a histogram of word occurrences as described above. We then reweight those histograms according to the TF-IDF measure [Salton and Buckley, 1988]. This reduces the weight on very common keywords. To learn each so-called weak classifier, we perform 50 iterations of bagging using C4.5.

Table 1 shows the results of these two methods and the standard dictionary method as measured by 3-fold cross-validation on the STONEFLY9 dataset. The results show that both the stacked random forest architecture and the boosted dictionary architecture achieve much better results than the standard dictionary method. A shortcoming of the stacked random forests is that the classifier cannot require multiple kinds of detections (e.g., both eyes and characteristic spots on the back). All detections just contribute to the total evidence for the classes, and the stacked classifier then finds an appropriate way to weigh the total evidence to make the final decision. In contrast, the dictionary methods are able to require multiple kinds of detections, as long as they correspond to multiple dictionary entries. Clearly, the boosted dictionary method is able to do a better job of this, because in the second and subsequent iterations, it is able to learn dictionaries that focus on the "hard" cases. One direction for future work is to combine boosting with the stacked random forests, as this may also allow the final classifier to require multiple kinds of detections in order to make a decision.

3 Automated Data Cleaning for Sensor Network Data

The second step in the Ecosystem Informatics pipeline is data cleaning. Ecology is one of the prime beneficiaries of the emerging technology of wireless sensor networks. We have been collaborating with the SensorScope project at the École Polytechnique Fédérale de Lausanne (EPFL) in Switzerland. SensorScope is a low-cost wireless platform for environmental sensor networks. In place of few, expensive permanent monitoring stations deployed sparsely over a large area, SensorScope allows field scientists to deploy many light-weight, inexpensive stations at a much higher spatial resolution. While SensorScope can support many different sensors, in our collaboration so far, we have studied only temperature sensors.

Although wireless sensor networks can be much less

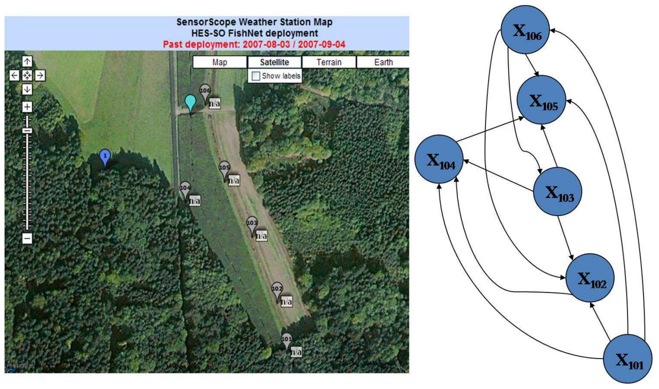

SensorScope Weather Station Map
HES-SO FishNet deployment
Past deployment: 2007-08-03 / 2007-09-04
Map
Satellite
Terrain
Earth
Show labels
X_{106}
X_{105}
X_{104}
X_{103}
X_{102}
X_{101}

Figure 4: Left: Top-down view of the FishNet Deployment. Right: Learned dependency relationships between the six sensor stations at the deployment.

expensive than traditional, permanent sensors, they are also subject to much more frequent failures. These can be caused by failure of the sensor, failure of the local network, or failure of the uplink from the field network to the research lab. The goal of our research is to develop automated data cleaning methods that can detect failures in real time and also fill in (impute) missing values to make the resulting data set more useful.

To achieve both of these functions, our approach is to learn a joint probability distribution over the outputs of all of the sensors. An advantage of these wireless sensor networks is that the larger number of sensors can provide redundancy that allows us to infer the true sensor value when a sensor fails.

Figure 4 shows a SensorScope deployment known as FishNet, where the variables $\{X_i\}$ indicate the true sensor values indexed by sensor i. Given a few weeks of sensor readings from this network, we assume that all sensors were working properly and learn a joint dynamic Bayesian network model that consists of three components. First, as shown in Figure 4(right), we learn the structure and parameters of a joint linear Gaussian model of the sensor readings. This network captures the correlations among the sensors at each time step. This becomes the "time slice" for the second component—a first-order dynamic Bayesian network in which there is a first-order linear Gaussian dependency between each variable in the network at time t and its value at time $t-1$. Finally, we introduce an observation model into the DBN such that the observed sensor value O_i depends on the true value X_i and a hidden sensor state variable S_i (where $S_i = 1$ means the sensor is working properly, in which case $O_i = X_i$ plus a *small* amount of Gaussian noise and where $S_i = 0$ means the sensor is broken, in which case $O_i = X_i$ but with a very *large* amount of Gaussian noise). All parameters of the model, except for the observation Gaussian noise, are learned from the data in the training period.

To perform data cleaning, we apply standard DBN filtering. At each time step, the observations are made, and we perform probabilistic inference to determine the most likely state of each sensor. We then assert this most likely state to be the true state, and assimilate the observations into the model to update the posterior distributions of the X_i variables at time t. If S_i is believed to be 0 at time t, the reading is marked as "faulty" and the posterior mean of X_i is used to predict (impute) the value for X_i.

Structure learning is accomplished by hill-climbing in the BGe metric of Geiger and Heckerman [1994]. Parameter learning for each conditional linear Gaussian is essentially linear regression via maximum likelihood.

Figure 5 shows the application of this method to days 22-41 of a deployment at Grand St. Bernard on the Swiss-Italian border. We can observe many sensor failures including bad sensor values (the spikes) and flat lines at -1 degrees, which are caused by network failures. The model was trained on the first 21 days of deployment and then applied to the remaining time. We can see that it is doing a very good job of identifying faults and imputing missing values.

4 Concluding Remarks

Computer science and artificial intelligence have the potential to transform the ecosystem sciences much the way they transformed molecular biology. Although many problems in ecology are superficially similar to previously-studied problems (e.g., object recognition, density estimation, model fitting, optimization), existing solutions are not directly applicable. This paper has shown one instance of this: standard methods for generic object recognition did not provide sufficient accuracy for recognizing stoneflies. Similarly, while predicting the distribution of a single species can be viewed as a Boolean classification problem, jointly predicting 5000 species poses a host of novel problems. For example, there are more than 12 million potential interactions among pairs of species. It is not feasible to estimate all of these interactions, even from large data sets. Another case arises with finding optimal policies for the active prevention of wildfires. In principle, these are just Markov decision problems, and we already have many methods for solving them. But if we consider a region made up of 10,000 management units, such that each year, we must choose 100 units on which to perform fuel reduction treatments, then our existing methods do not scale. This problem has $O(10^{500})$ potential actions at each time point!

To prepare students to work in ecosystem informatics, we have created two educational programs. Each summer, we conduct a 10-week Summer Institute in Eco-Informatics at the H. J. Andrews Experimental Forest (`eco-informatics.engr.oregonstate.edu`. This provides an opportunity for juniors, seniors, and first-year graduate students to work in interdisciplinary teams that combine field work with mathematical and computational modeling. We have also established an interdisciplinary graduate program in Ecosystem Informatics under the NSF IGERT program (`ecoinformatics.oregonstate.edu`). This program trains students to conduct research in teams that combine mathematics, computer science, and the ecosystem sciences. Students receive rigorous education in their core discipline while also obtaining a Ph.D. minor in Ecosystem Informatics.

I urge everyone to join us in addressing these interesting research problems. Given the ecological challenges facing our planet, there is an urgent need to develop the underlying science that can guide policy making and implementation. Ecology is poised for a data-driven revolution that can help it address these needs. But ecologists can't do this alone. They need computer scientists to accept the challenge and develop the novel computational tools that can make this revolution a reality.

Acknowledgments

The author gratefully acknowledges the support of NSF grants IIS-0326052, IIS-0705765, and DGE-0333257. The author thanks the BugID team for their contributions to the arthropod recognition results and Ethan Dereszynski for the data cleaning work described here.

References

[Breiman, 2001] L. Breiman. Random forests. *Mach. Lrn.*, 45(1):5, 2001.

[Deng *et al.*, 2007] H. Deng, W. Zhang, E. Mortensen, T. Dietterich, and L. Shapiro. Principal curvature-based region detector for object recognition. In *CVPR2007*, pages 1–8, 2007.

[Elith *et al.*, 2006] J. Elith, C.H. Graham, and et al. Novel methods improve prediction of species' distributions from occurrence data. *Ecography*, 29:129–51, 2006.

[Freund and Schapire, 1997] Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, 55(1):119–39, 1997.

[Geiger and Heckerman, 1994] D. Geiger and D. Heckerman. Learning Gaussian networks. Technical Report MSR-TR-94-10, Microsoft Research, 1994.

[Kadir and Brady, 2001] T. Kadir and M. Brady. Scale, saliency and image description. *IJCV*, 45(2):83–105, 2001.

[Kalal *et al.*, 2008] Z. Kalal, J. Matas, and K. Mikolajczyk. Weighted sampling for large-scale boosting. In *BMVC*, 2008.

[Krause *et al.*, 2008] A. Krause, A. Singh, and C. Guestrin. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *JMLR*, 9:235–84, 2008.

[Larios *et al.*, 2008] N. Larios, H. Deng, W. Zhang, M. Sarpola, J. Yuen, R. Paasch, A. Moldenke, D. Lytle, S. Ruiz Correa, E. Mortensen, L. Shapiro,

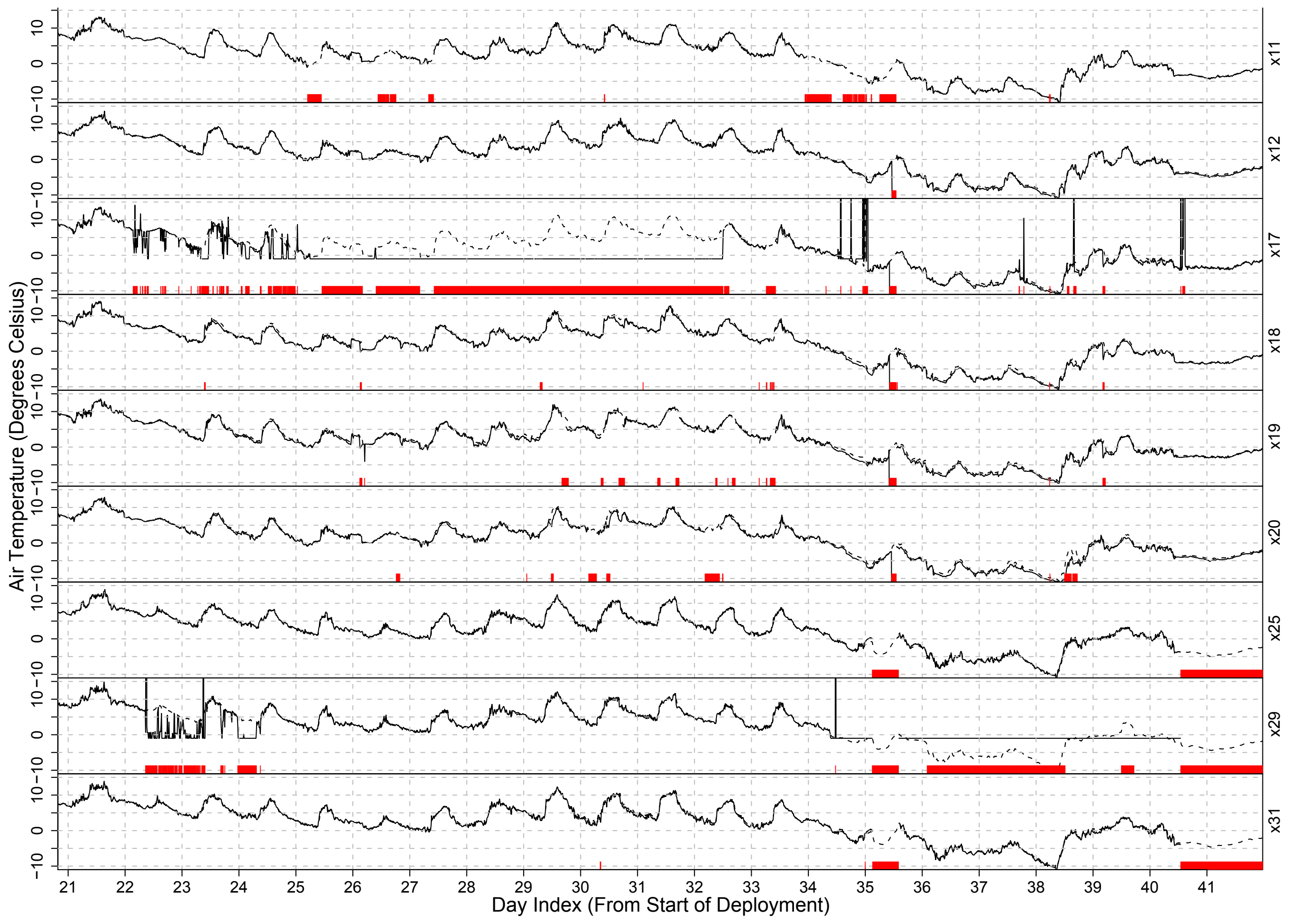


Figure 5: Quality Control performance on Grand St. Bernard. Solid line indicates the actual temperature recorded at each station. Dashed line indicates the posterior prediction made for that station. Red hashes indicate values labeled as "faulty". The X axis denotes the day since the deployment began, and the Y axis denotes temperature in degrees.

and T. Dietterich. Automated insect identification through concatenated histograms of local appearance features. *Machine Vision and Applications*, 19(2):105–123, 2008.

[Lowe, 2004] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.

[Martínez-Muñoz *et al.*, 2009] G. Martínez-Muñoz, W. Zhang, and et al. Dictionary-free categorization of very similar objects via stacked evidence trees. In *CVPR 2009*, 2009.

[Mikolajczyk and Schmid, 2002] K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. In *ECCV*, pages 128–42, 2002.

[Moosmann *et al.*, 2007] F. Moosmann, B. Triggs, and F. Jurie. Fast discriminative visual codebooks using randomized clustering forests. In *NIPS 19*, pages 985–92. 2007.

[Salton and Buckley, 1988] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–23, 1988.

[Winn *et al.*, 2005] J. Winn, A. Criminisi, and T. Minka. Object categorization by learned universal visual dictionary. In *ICCV*, volume 2, pages 1800–07, 2005.

[Yang *et al.*, 2008] L. Yang, R. Jin, R. Sukthankar, and F. Jurie. Unifying discriminative visual codebook generation with classifier training for object category recognition. In *CVPR*, 2008.

[Zhang and Dietterich, 2008] W. Zhang and T. Dietterich. Learning visual dictionaries and decision lists for object recognition. In *ICPR2008*, pages 1–4, 2008.

[Zhang *et al.*, 2009] W. Zhang, X. Fern, and T. G. Dietterich. Learning non-redundant codebooks for classifying complex objects. In *ICML 2009*, 2009.