Machine Learning for Earth Observation Flight Planning Optimization

Elif Kürklü[1,2] and **Robert A. Morris**[2] and **Nikunj Oza**[2]
(1) Perot Systems Government Services
(2) Intelligent Systems Division
NASA Ames Research Center

Abstract

This paper is a progress report of an effort whose goal is to demonstrate the effectiveness of automated data mining and planning for the daily management of Earth Science missions. Currently, data mining and machine learning technologies are being used by scientists at research labs for validating Earth science models. However, few if any of these advanced techniques are currently being integrated into daily mission operations. Consequently, there are significant gaps in the knowledge that can be derived from the models and data that are used each day for guiding mission activities. The result can be sub-optimal observation plans, lack of useful data, and wasteful use of resources. Recent advances in data mining, machine learning, and planning make it feasible to migrate these technologies into the daily mission planning cycle. This paper describes the design of a closed loop system for data acquisition, processing, and flight planning that integrates the results of machine learning into the flight planning process.

Introduction

Machine learning has been integrated with planning systems in order to automatically extract knowledge from one problem to apply to future problems, thereby improving the performance of the planning system. More specifically, knowledge about the domain, search control strategies, or solution quality, can often be refined or extended from the ”experience” of planning.

The work described in this paper fits into the overall idea of ”learning to improve planning” but with an emphasis that distinguishes it from the goals of other efforts. Specifically, the ”experience” that improves the planning is gained from (data acquired as the result of) the *execution* of previous instances of the planning problem, or from other sources, rather than the experience of planning (i.e. the search for a plan) itself. The application here is the identification of useful or interesting observation targets for an in-situ sensor mounted on an airborne observatory (a modified DC-8). Typically, targets are interesting if they reveal discrepancies or anomalies in predictive models used by Earth scientists to study processes related to things like climate change or pollution. Further observations of such targets will quantify the errors in the predictive models in order to improve their predictive accuracy or more generally fill gaps in the knowledge about the process of interest. Interesting observation targets become goals used by a flight planner to generate the next day's observation activities. Furthermore, because the process under investigation (e.g. a hurricane or pollution plume) may be changing daily, a *cycle* of planning, observation, analysis of data acquired from observation, and model revision and prediction, occurs continuously throughout the course of an *observation campaign* (typically lasting weeks).

This paper presents the design of a system for integrating planning and learning capabilities for use by human mission planners to improve the daily operations of what are called *mixed observation* missions, i.e., missions that combine observations from a number of heterogeneous ground, airborne, or space-borne sensors. The next section offers an example of such a mission and motivates the automation. There follows sections on the learning and planning components, and how they will be integrated into a system that assists the human operator in the observation management cycle. The paper closes with a discussion of broader issues that could be addressed by automated planning and learning.

Background and Motivation

NASA and other international space agencies launch and operate Earth observing systems for collecting remote sensing measurements to support scientists in the pursuit of goals related to understanding changes to the Earth's ecosystem. These data are combined with data collected from *in-situ* sensors mounted on airborne or ground platforms. Many scientific goals require the combination of data acquired from different sensors. For example, to expand the temporal and spatial scale of airborne measurements, measurements from remote sensors are obtained simultaneously with in-situ sensors.

An example of such a mixed platform observation mission was INTEX-B (INTEX-B), conducted from March 1 to April 30, 2006. INTEX-B was the second part of a two-phase experiment that aimed to understand the transport and transformation of gases and aerosols on transcontinental and intercontinental scales and assess their impact on air quality and climate. INTEX-B science goals include quantification of the outflow of pollutants and aerosols from North America over the Atlantic to Europe; an improvement of our understanding of the chemical and physical evolution of these constituents; and an assessment of the impact of pollution transported from mega-cities such as Mexico City. High-priority measurements included long-lived greenhouse gases, ozone and its precursors, and aerosols and their precursors.

Day-to-day DC-8 flight planning on INTEX-B was a process of generating a set of flight legs between waypoints (the flight plan). Plan generation was partly automated using various flight planning tools that assist the human user by accounting for flight dynamics of the DC-8 and other constraints. Selection of waypoints was guided by forecasts generated from a set of models of different spatial scales. Near-real-time observations from a number of satellite instruments were used to guide the selection of and to identify specific regions of interest for in-situ sampling. Integration of aircraft and satellite measurements to address the mission objectives requires validation flights directed at establishing the consistency between the two data sets.

The focus of automation in the context of mixed observation missions such as INTEX is to improve the ability to *assimilate* the different data sources in order to generate waypoints; and secondly, to automate the *search* for high quality plans, where quality is related to the ability to resolve model anomalies or accomplish other mission goals. The idea is that improved data assimilation will result in a better understanding of the space of high quality flight plans, a space which can be then explored more effectively by automated planning methods. The next sections investigate the approaches used for each of these capabilities.

Architecture

The semi-closed system integrating machine learning with flight planning into a daily mission planning system is visualized in Figure 1. The closed loop system involves four components:

1. Data archives, process models, and instruments for producing data or forecast products.
2. A *data assimilation assistant* for generating planning goals (waypoints) given data;
3. A *flight and activity planning assistant* for generating flight plans given goals; and
4. A *visualization tool* VIZ (Edwards *et al.* 2004) for displaying the data and plans to the human flight planner.

In the current implementation, a process model produces Carbon Monoxide (CO) distribution predictions, which are assimilated with satellite observations of CO from remote sensors such as Aurora and Ionospheric Remote Sensing (AIRS) and Measurements of Pollution in the Troposphere (MOPITT) sensors, as well as observations of dust, aerosols and clouds from the Moderate Resolution Imaging Spectroradiometer (MODIS). These data and models are organized and archived in the the Terrestrial Observation and Prediction System (TOPS) data and modeling system (Nemani *et al.* 2000). Waypoints, currently expressed in a 3D location vector, $\langle latitude, longitude, altitude \rangle$ are identified as interesting locations for conducting future CO measurements, either for finding and characterizing differences between models and observations, or for filling in gaps in data.

The waypoints are fed into the planner as goals. Flight plans are generated using the greedy approach described above, and the best plan displayed to the human planner. The remainder of this paper describes the learning and flight planning approaches utilized.

Learning Problem

The high-level learning problem in this paper is the problem of learning to map measurements taken as part of a mission with other relevant data and physical process models to scientifically useful locations/times where new measurements should be taken. Within this high-level learning problem, there are several lower-level learning problems that we are investigating:

1. Physical process models such as MOZART (Brasseur *et al.* 1998a; 1998b) represent expert knowledge. Situations in which these models make unexpected or erroneous predictions are situations that need to be studied further, since expert knowledge is unable to explain them. For example, MOZART generates predictions of Carbon Monoxide (CO). Several satellite instruments such as MOPITT and AIRS as well as in-situ instruments onboard the DC-8 yield CO measurements. Machine learning and data mining methods can be used to model the errors made by MOZART as a function of the available measurements, measurement locations, and/or background conditions (such as temperature and pressure). The methods would generate models based on past data and then would be applied to forecasts for the next day to generate predictions of where the errors are likely to be highest (e.g., locations, temperature ranges). These represent suggestions of measurements to be done or re-done as well as how the physical models themselves may be modified. The resulting data mining/machine learning models can also be used to fill in the gaps left by the physical models. That is, one can leave the existing physical models as they are and add our data mining models to them to form an ensemble model that outperforms the original physical models.

2. Just as physical models and measurements may be inconsistent with each other, different measurements and data may be inconsistent. Machine learning and data mining methods can be used to model the relevant measurements and other data and can use the resulting models to detect such inconsistencies in new data. These inconsistencies represent anomalous conditions under which one may want to take measurements again. This capability allows us to go far beyond typical single-variable range-checking and instead enables the detection of deviations from the typical mathematical relationships between measurements and data even when they may be within appropriate ranges.

3. Measurements and derived data that are needed for flight planning may sometimes be missing or corrupted. This may happen due to sensor problems or conditions that hamper measurement (e.g., clouds). Certain data derived from measurements and process models may also be useful in flight planning but may not be available due to the

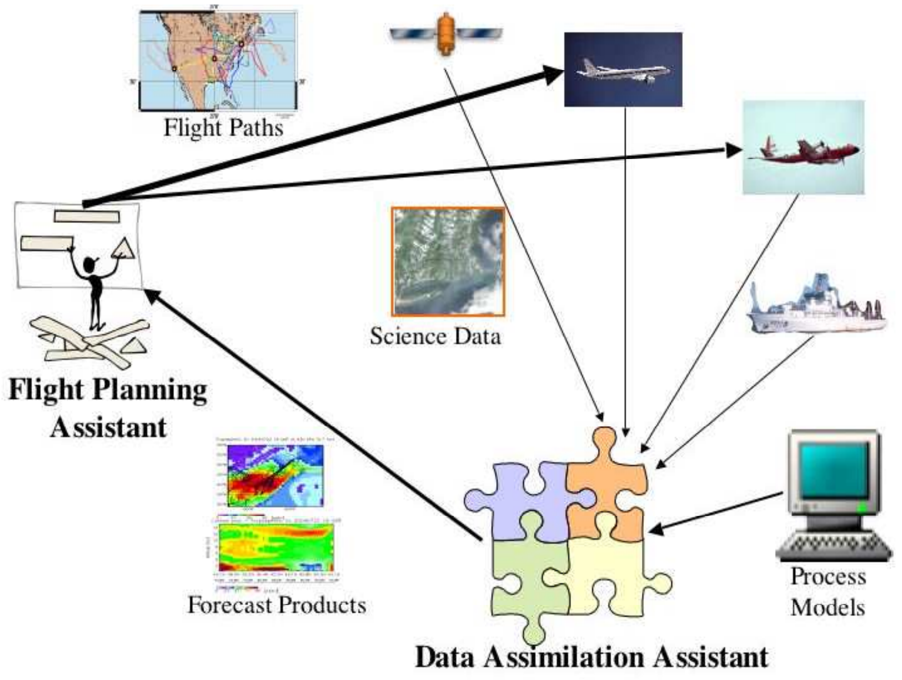

Figure 1: The daily mission flight planning cycle, integrating machine learning and flight planning.

time required to compute the derived data. Data mining and machine learning methods can generate estimates of sensor measurements and derived data as a function of other available data and information in situations where true values are unavailable or unusable.

For these three functions, we are using standard machine learning methods as well as novel methods developed by us and our colleagues (Srivastava 2004; Iverson 2004; Bay & Schwabacher 2003; Oza 2004). We are using supervised machine learning methods for regression such as MultiLayer Perceptrons (MLPs) (Bishop 1995) and Support Vector Regression (SVR) with Radial Basis Function (RBF) kernels (Scholkopf & Smola 2002). To use these methods, we identify a training set consisting of pairs $\{z_1, y_1), (z_2, y_2), \ldots, (z_n, y_n)\}$, where n is the number of training examples we have, and each training example consists of a quantity y (the output or response) that we would like to predict and z (the input or predictor) that we believe is useful in generating the output. For example, in low-level problem 1 above, each y_i ($i \in \{1, 2, \ldots, n\}$)could be the error in MOZART's CO prediction in a certain region and z could be the actual MOZART CO prediction and temperature. The machine learning model would learn a function $y = f(z)$ from past data. Then, given the CO prediction and temperature forecast for the next measurement day, the machine learning model would generate an estimate of MOZART's prediction error. The high-error regions would represent locations where measurements should be taken. In problem 2 above, we are using the nonlinear regression methods just described to model different measurements and data as a function of other available data. Such nonlinear models go far beyond simply measuring the linear correlations between different data and checking for changes.

We are also using standard unsupervised learning methods for clustering (e.g., K-means (MacQueen 1967) and kernel clustering (Girolami 2001)) and anomaly detection (e.g., one-class Support Vector Machines (Tax, Ypma, & Duin 1999)). Clustering methods organize the data into clusters that represent data that have similar properties (e.g., particular ranges in which different measurements fall or particular mathematical relationships between measurements). Data that are part of very small clusters or are far from all the clusters represent possible anomalies that may need to be assessed. Anomaly detection methods construct a model of the training data, which is typically assumed to be normal. These methods are then executed on new data, and data that do not fit the model's definition of normality are flagged as anomalous. For example, one-class Support Vector Machines (SVMs) are given training inputs $z_1, z_2, \ldots, z_n$ just as defined above, as well as an expected fraction of training points that are anomalous. One-class SVMs find a nonlinear model that identifies up to that fraction of training points as anomalous and separates them from the normal points. When a one-class SVM is given a training point z, it returns a value y that is positive if z is part of the normal regime identified in the training data and negative if z is thought to be abnormal. The farther away y is from zero, the "more normal/abnormal" z is. Given input from clustering and anomaly detection, we expect to characterize regions where new measurements should be taken (e.g., particular locations or ranges of model predictions). We expect that different methods for supervised learning, clustering, and anomaly detection will work well in different situations (e.g., different times of year or regions of the Earth). Therefore, we plan to use ensemble machine learning methods (Kuncheva 2004) that combine multiple machine learning models in order to leverage each model's strengths in the right situations.

For the third function above, we are leveraging work done under the Virtual Sensors project (Srivastava, Oza, & Stroeve 2005). The goal of this project was to produce high accuracy estimates of sensor measurements given other measurements and data. These estimates are produced when actual measurements are not available, which can happen due to sensor failure, ambient conditions that prevent or hinder measurement, or the measurement capability not being available (e.g., for an older instrument). The estimator is referred to as a Virtual Sensor because it is designed to serve as an estimate for a real sensor measurement when the real measurement is not available. Virtual Sensors exploit the fact that, even though every sensor takes seemingly independent measurements, there are a limited set of possible objects that these sensors measure. Therefore, all the sensors cannot simultaneously take all possible values (e.g., two sensors taking CO measurements in the same location will tend to report comparable values, therefore, if one sensor is unavailable, the other one can be used to generate an estimate of the missing measurement and/or a range of possible measurements). We previously used this methodology to generate an estimate of the Moderate Resolution Imaging Spectroradiometer (MODIS) channel 6 (1.6 mm) for an older instrument (Advanced Very High Resolution Radiometer, AVHRR/2) that did not have this channel (Srivastava, Oza, & Stroeve 2005).

In this work, instead of using supervised and unsupervised learning methods on the data alone, we could have used inverse reinforcement learning (e.g., (Abbeel & Ng 2004)), which would learn a utility function from past examples of good plans, and use that utility function to generate new plans in the future. This would be a more direct approach, since it would avoid explicitly representing differences or errors between measurements and/or process models. However, we chose our approach because the resulting models could be used to augment existing physical models as described in learning problem 1 above. Additionally, the measurement errors that our models predict would be more directly interpretable by domain experts than plans would be. The measurement errors can be seen as a plan selection rationale, which would be critical for domain expert acceptance of our planned system.

Planning Problem

There are two kinds of inputs to the flight planner: mission goals and waypoints. Mission goals arise from models, from a phenomenon or event of interest, or from other sensing resources. The goal might be to validate predictions made by a model, to characterize or classify the composition of dynamic process like a pollution plume, or validate observations made by a remote sensing instrument. The waypoints are 3D specifications of locations that support one or more of the goals. Goals or waypoints may be assigned a priority. Eventually (although not in the planning system described in this section) waypoints may also be assigned a indicator of confidence or certainty.

The output of the flight planner is a flight plan, a closed path comprised of a set of segments, or flight legs, connecting waypoints. A path is also associated with temporal information (start time of each leg, and duration). Each leg follows a pattern: direct ascent, direct descent, level, or spiral (ascent or descent). The best path is the one the planner has determined most likely to satisfy the most goals, based on the specified priorities.

There are four kinds of constraints on solutions generated by the planner: related to the instruments taking the measurements, the platforms (aircraft) on which they reside, direct constraints on the paths that can be flown, or the mission goals.

First, nstruments must be set up before they can take measurements. In this problem, to set up an instrument means setting up the aircraft on which it resides. This means two things: for the aircraft to assume a certain flight pattern, and to remain in this pattern for a specified duration of time. For example, to take a measurement on an in-situ instrument might require level flying for at least 10 minutes.

Second, the aircraft on which the instruments reside have requirements related to its navigational capabilities, or for safe flying. The flight path must adhere to these requirements. An important constraint of this type is the avoidance of Special Use Airspace (SUA).

Third, the paths flown must satisfy certain constraints, such as the flight path must start and end at the same location, the airport at which the aircraft resides. Restricted airspace requirements are also path constraints. There may be constraints related to start times or end times of the flight plan.

Finally, goals constrain flight plans by requiring certain patterns. Thus, the goal to validate a remote sensing instrument requires a flight plan that contains legs that coincide with the flight pattern of the satellite. Another example from INTEX is that to take measurements at low altitudes the DC-8 must simulate a ”missed approach” to an airfield, i.e., a maneuver in which the plane overshoots a landing field during descent and must ascend to a cruising altitude.

The core computational problem being solved can be viewed as a version of the *orienteering problem* (OP) (Fischetti, Gonzalez, & Toth 1998). In this type of transportation problem there is a single ”vehicle” (the DC-8) and a set of ”customers” (waypoints) that need to be serviced, each with a designated priority. There is a transit time between customers. A feasible route is a sequence of waypoints that begins and ends at a designated ”depot” (airport). A feasible route also must adhere to a bound on the sum of the transit times in the route, corresponding to restrictions on fuel and other restrictions on the length of a flight. More formally: given a set of waypoints, $W = \{x_0, x_1 ... x_n\}$ each with an assigned priority $w_i, 1 \leq i \leq n$, a binary cost function $C : W \times W \rightarrow N$, a designated start point x_0 with $w_0 = 0$, and a temporal bound B, find a sequence (schedule) $s = \langle x_{s_1}, ..., x_{s_k} \rangle, s_j \in \{0, \ldots, n\}, k \leq n + 1$, that maximizes $\Sigma_{j=1 \ldots k} w_{s_j}$, subject to the following constraints:

1. $s_1 = s_k = 0$,
2. $i \neq j \rightarrow s_i \neq s_j$, i.e., each $x_j \in W$ occurs at most once in s,
3. $\Sigma_{i=2 \ldots k} C(x_{s_{i-1}}, x_{s_i}) \leq B$.

Notice that in this version there is an assumption of ”over-subscription”: it is not necessary, and may not be possible, to service all the customers, and indeed a best route might not include all of them. As such, it bears similarity to the robot activity planning problem described in (Smith 2004). The solution method employed here also draws upon an approach to flight planning employed in the Sofia planning system (Frank, Gross, & Kürklü 2004).

In the first version of the solver (flight planner), we are experimenting with variations of a constructive search, where each decision point involves selection of the next waypoint to add to a partial schedule. A greedy approach to selection is employed, where each feasible waypoint candidate x_i is heuristically evaluated in terms of the *expected value* v_i of adding a leg terminating at x_i to the partial plan. We define $v_i = w_i/[C(x_{s_j}, x_i) + p_{j,i}]$, where x_{s_j} was the last waypoint added to the schedule, and $p_{j,i}$ is a penalty for SUA intrusion, discussed below. A candidate x_i is feasible for a partial schedule $\langle x_{s_1}, \ldots, x_{s_j} \rangle$ if the sequence $\langle x_{s_1}, \ldots, x_{s_j}, x_i, x_{s_1} \rangle$ satisfies the bound constraint (3), i.e., if the aircraft can return to the airport immediately after flying to the waypoint without violating (3). The algorithm deterministically selects the candidate with the highest expected value to extend a plan, until either the list of available candidates is empty, or none of the remaining candidates is feasible with respect to the current plan.

SUA intrusion is treated as a penalty $p_{j,i}$ that reduces the expected value of adding a leg from x_{s_j} to x_i to the current plan. The penalty is incurred because for the leg to be added to the plan, it is necessary to plan a route around the SUA. Because the new path adds duration to the overall plan, the candidate loses value proportionately.

An SUA violation is detected by intersecting the line defined by the leg with the region defining the SUA. More than one SUA may be defined for a given problem, so this intersection may need to be performed more than once for each candidate. If the intersection is non-empty, then the value of the penalty, $p_{j,i}$, must be estimated. A number of estimation techniques are available. We start with one of the simplest, which does not involve any form of path planning. Given a SUA region, we compute its convex hull using the Graham Scan algorithm, which has complexity $O(n \log n)$, and determine the perimeter of the resulting hull. We estimate the worst-case penalty $p_{j,i}$ as one-half the perimeter. This corresponds to the scenario where, roughly speaking, the straight line path intersects the ”mid point” of the polyhedron. If more than one SUA is intruded upon, then we perform the same estimate on the convex hull of the union of the relevant SUAs.

The flight planning algorithm can be described as follows:

```
Flight Planner
While there are observations left to schedule
    1. Greedily select next observation to schedule:
            For each unscheduled observation
                Evaluate cost of adding the observation to
                the schedule, including the need to avoid SUAs.
    2. Add enabling leg
            Check altitude constraints
            Find path to avoid SUA incursions, if necessary
            Check to ensure resulting schedule does not violate
            flight duration constraints.
    3. Add observation leg (same steps as in 2.)
Add a flight leg to return to origin.
Return Plan
```

For each waypoint in the input to the problem, the planner computes the expected value v_i, described above, and greedily selects the observation with the highest such value. An *enabling leg* is added to the plan, which is the shortest path to get to the observation point. The leg must not violate altitude, duration, or SUA incursion constraints. In this phase, each SUA along the path must be examined. Again, its convex hull is computed, and a *visibility graph* is constructed out of the vertices of all the hulls. Dijkstra’s shortest path algorithm (complexity $O(n^2)$) is then applied to find the shortest cost path around the SUAs Note that as a result of this approach, new waypoints may be added to the flight plan, corresponding to the subset of the vertices of the SUA convex hulls which were part of the shortest path. An *observation leg* is then added to the plan, corresponding to the time of the measurement at the desired location. The same tests for feasibility are made as those made when the enabling leg is added.

The worst case performance of the algorithm is polynomial in a number of different parameters, including the number of observations to be scheduled, the number of SUAs, and the maximum number of vertices in the visibility graphs constructed out of the SUA convex hulls (which is a factor of the shape of the SUA). An analysis of the planner suggests that the main source of computational overhead is the SUA avoidance. This analysis is borne out by experiments. Future reports will summarize these results.

Conclusion

This work is an example of the infusion of AI technology into decision support tools for remote sensing missions for Earth science. The integration of machine learning and data mining into mission observation planning will increase the amount of useful data products for improving the predictive capabilities of Earth science models, thus improving human understanding of Earth processes. From a technology standpoint, the use of observation and forecast data in the formulation of planning goals represents a more robust representation of the world in which the plans will be executed, which will improve the ability of planning systems to converge on plans with high scientific value.

References

Abbeel, P., and Ng, A. 2004. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the Twenty-first International Conference on Machine Learning*.

Bay, S. D., and Schwabacher, M. 2003. Mining distance-based outliers in near linear time with randomization and a simple pruning rule. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Bishop, C. 1995. *Neural Networks for Pattern Recognition*. New York: Oxford University Press.

Brasseur, G.; Hauglustaine, D.; Walters, S.; Rasch, P.; Muller, J.; Granier, C.; and Tie, X. X. 1998a. Mozart: a global chemical transport model for ozone and related chemical tracers, part 1. model description. *Journal of Geophysical Research* 103:28,265–28,289.

Brasseur, G.; Hauglustaine, D.; Walters, S.; Rasch, P.; Muller, J.; L. K., E.; and Carroll, M. A. 1998b. Mozart: a global chemical transport model for ozone and related chemical tracers, part 2. model results and evaluation. *Journal of Geophysical Research* 103:28,291–28,335.

Bresina, J. L. 1996. Heuristic-biased stochastic sampling. In *AAAI/IAAI, Vol. 1*, 271–278.

Edwards, L.; Bowman, J.; Kunz, C.; Lees, D.; and Sims, M. 2004. Photo-realistic terrain modeling and visualization for mars exploration rover science operations. In *IEEE SMC 2005*. Proceedings of the 16th Conference on Innovative Applications of Artificial Intelligence.

Fischetti, M.; Gonzalez, J. J. S.; and Toth, P. 1998. Solving the orienteering problem through branch-and-cut. *INFORMS Journal on Computing* 10:133–148.

Frank, J.; Gross, M.; and Kürklü, E. 2004. Sofia's choice: an ai approach to scheduling airborne astronomy observations. In *Proceedings of IAAI*.

Girolami, M. 2001. Mercer kernel based clustering in feature space. *IEEE Transactions on Neural Networks* 13:780784.

INTEX-B. http://www.espo.nasa.gov/intex-b/.

Iverson, D. L. 2004. Inductive system health monitoring. In *Proceedings of the International Conference on Artificial Intelligence*, volume 2.

Kuncheva, L. I. 2004. *Combining Pattern Classifiers*. New Jersey: Wiley-IEEE.

MacQueen, J. B. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability*, volume 1. Berkeley: University of California Press. 281–297.

Nemani, R.; Votava, P.; Roads, J.; White, M.; Thornton, P.; and Coughlan, J. 2000. Terrestrial observation and prediction system: Integration of satellite and surface weather observations with ecosystem models. In *Proceedings of the 4th International Conference on Integrating GIS and Environmental Modeling (GIS/EM4)*. Banff, Canada.

Oza, N. C. 2004. Aveboost2: Boosting for noisy data. In Kittler, J.; Roli, F.; and Windeatt, T., eds., *Proceedings of the Fifth International Workshop on Multiple Classifier Systems*. Berlin: Springer. 31–40.

Scholkopf, B. ., and Smola, A. 2002. *Learning with Kernels*. Cambridge: MIT Press.

Smith, D. E. 2004. Choosing objectives in over-subscription planning. In *Proceedings of the 14th International Conference on Automated Planning and Scheduling , (ICAPS 2004)*.

Srivastava, A. N.; Oza, N. C.; and Stroeve, J. 2005. Virtual sensors: Using data mining to efficiently estimate spectra. *IEEE Transactions on Geosciences and Remote Sensing, Special Issue on Advances in Techniques for Analysis of Remotely Sensed Data* 43:590?600.

Srivastava, A. 2004. Mixture density mercer kernels: A method to learn kernels directly from data. In *Proceedings of the SIAM Data Mining Conference*. Society for Industrial and Applied Mathematics.

Tax, D. M. J.; Ypma, A.; and Duin, R. P. W. 1999. Pump failure detection using support vector data descriptions. In *Lecture Notes in Computer Science*, volume 1642. 415+.