Monitoring the Errors of Discriminative Models with Probabilistic Programming

Christina Curlette  
Probabilistic Computing Project  
Massachusetts Institute of Technology  
curlette@mit.edu

Ulrich Schaechtle  
Probabilistic Computing Project  
Massachusetts Institute of Technology  
ulli@mit.edu

Vikash K. Mansinghka  
Probabilistic Computing Project  
Massachusetts Institute of Technology  
vkm@mit.edu

Machine learning algorithms produce predictive models whose patterns of error can be difficult to summarize and predict. Specific types of error may be non-uniformly distributed over the space of input features, and this space itself may be only sparsely and non-uniformly represented in the training data. This abstract shows how to use BayesDB, a probabilistic programming platform for probabilistic data analysis, to simultaneously (i) learn a discriminative model for a specific prediction of interest, and (ii) build a non-parametric Bayesian generative “monitoring model” that jointly models the input features and the probable errors of the discriminative model. Because it is hosted in a probabilistic programming system, the generative model can be interactively queried to (i) predict the pattern of error for held-out inputs, (ii) describe probable dependencies between the error pattern and the input features, and (iii) generate synthetic inputs that will probably cause the discriminative model to make specific errors. Unlike approaches based purely on optimizing error likelihood — including recently proposed approaches for finding “optical illusions” for neural nets — the generative monitor also accounts for the typicality of the input under a generative model for the training data. This biases synthetic results towards plausible feature vectors. Figure 1 shows a schematic of the overall approach. This abstract illustrates these capabilities using the problem of classifying the orbits of Earth satellites. The underlying data source is maintained by the Union of Concerned Scientists; three representative satellites are shown in Table 1.

An autonomous system that makes predictions for self-driving cars or makes decisions in a medical context cannot function as a black box; rather, it needs to be explainable and reliable (e.g. [Holdren and Smith, 2016] [DARPA, 2016]. Many recent research efforts aim to understand what impairs the reliability of machine learning systems and how they can be improved. [Hand, 2006] pointed out that some common assumptions made when using discriminative, predictive models render them unreliable. For example, training data is often not uniformly drawn from the distribution of the data that the predictive model will be applied to. Additionally, false certainty about the correctness of the training data’s labels can further lead to suboptimal performance. [Sculley et al., 2014] explain such suboptimal performance as a kind of technical debt caused by a machine learning system’s dependence on training and test data. The authors describe the ability to monitor a machine learning system as “critical.” Some recent work has focused on building monitoring models for machine learning systems; for example, [Ribeiro et al., 2016] introduced a monitoring system that infers explanations by observing and analyzing the input-output behavior of an arbitrary opaque predictive model.

Our approach, outlined in Figure 1, makes use of probabilistic programming to jointly model input features and the types of errors made by discriminative black box machine learning models, which we call error pattern. We define error pattern as a function of the true class of an input and the predicted class of an input, whose output could be, e.g., a tuple of true class and predicted class or a Boolean value representing whether the predicted class is correct. One key component of our system is BayesDB, a probabilistic programming platform for probabilistic data analysis [Mansinghka et al., 2015b]. A second key component is CrossCat, a Bayesian non-parametric method for learning the joint distribution over all variables in a heterogeneous, high-dimensional population [Mansinghka et al., 2015a].

Presented at the Workshop on Reliable Machine Learning in the Wild (NIPS 2016), Barcelona, Spain.
Figure 1: **Reliability analysis with probabilistic programming.** (a) BayesDB’s Meta-Modeling Language can be used to train a generative monitoring model. A generative monitoring model is a model for the joint distribution over input features, annotations, and error patterns. A black box machine learning (ML) algorithm is trained using a set of training data \( \{(\vec{x}_i, y_i)\} \), resulting in a predictive model \( f_\theta(\vec{x}) \) where \( \theta \) is a set of parameters for the predictive model. Monitoring data consist of input features \( \vec{x} \), a potentially sparse auxiliary signal \( \vec{\alpha} \), and error pattern \( \varepsilon \) as a function of the predictive model and the true target. We model the generative monitor \( P_\phi(\vec{x}, \vec{\alpha}, \varepsilon) \) using CrossCat, where \( \phi \) denotes the structure and parameters for CrossCat. (b) BayesDB’s Bayesian Query Language (BQL) can be used to query the generative monitoring model in order to explain error patterns, predict likely failures, and predict the likelihood that error pattern depends on other variables using mutual information; which above is written as \( I(\vec{x}; \varepsilon) \).

Table 1 on the next page shows three representative entries in the dataset of satellites used for the monitoring model example in this abstract.

In the example considered in this abstract, a black box random forest model is used to predict satellites’ type of orbit. The random forest model takes in a set of training data, fits decision tree classifiers on subsamples of the data, and uses an aggregate of the decision trees’ classifications to make predictions. For a size \( n \) set of training data \( \{(\vec{x}_i, y_i)\} \) and a total of \( B \) decision trees, \( n \) samples are drawn with replacement \( B \) times. Each set of samples drawn on step \( b = 1 \ldots B \) is used to train a decision tree \( f_b \) using a standard tree learning algorithm. At each split in tree \( f_b \), a random subset of \( \sqrt{n} \) of the total \( n \) features are used as predictors. The random forest then classifies a new input \( \vec{x}_j \) using the majority of the decision trees’ classifications, i.e. mode(\( f_b(\vec{x}_j) \)) for \( b = 1 \ldots B \). In this case, the random forest’s predictions are used to create a variable called classification error pattern, a string
Table 1: **Satellites data table** (Saad and Mansinghka 2016). Variables in the satellites population and three representative satellites. The records are multivariate, heterogeneously typed, and contain arbitrary patterns of missing data.

<table>
<thead>
<tr>
<th>Name</th>
<th>International Space Station</th>
<th>AAUSat-3</th>
<th>Advanced Orion 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Country of Operator</td>
<td>Multinational</td>
<td>Denmark</td>
<td>USA</td>
</tr>
<tr>
<td>Operator Owner</td>
<td>NASA/Multinational</td>
<td>Aalborg University</td>
<td>NRO</td>
</tr>
<tr>
<td>Users</td>
<td>Government</td>
<td>Civil</td>
<td>Military</td>
</tr>
<tr>
<td>Purpose</td>
<td>Scientific Research</td>
<td>Technology Development</td>
<td>Electronic Surveillance</td>
</tr>
<tr>
<td>Class of Orbit</td>
<td>LEO</td>
<td>LEO</td>
<td>GEO</td>
</tr>
<tr>
<td>Type of Orbit</td>
<td>Intermediate</td>
<td>NaN</td>
<td>NaN</td>
</tr>
<tr>
<td>Perigee km</td>
<td>401</td>
<td>770</td>
<td>35500</td>
</tr>
<tr>
<td>Apogee km</td>
<td>422</td>
<td>787</td>
<td>35500</td>
</tr>
<tr>
<td>Eccentricity</td>
<td>0.00155</td>
<td>0.00119</td>
<td>0</td>
</tr>
<tr>
<td>Period minutes</td>
<td>92.8</td>
<td>100.42</td>
<td>0</td>
</tr>
<tr>
<td>Launch Mass kg</td>
<td>NaN</td>
<td>8.0</td>
<td>5000</td>
</tr>
<tr>
<td>Dry Mass kg</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
</tr>
<tr>
<td>Power watts</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
</tr>
<tr>
<td>Date of Launch</td>
<td>36119</td>
<td>41330</td>
<td>40503</td>
</tr>
<tr>
<td>Anticipated Lifetime</td>
<td>30</td>
<td>1</td>
<td>NaN</td>
</tr>
<tr>
<td>Contractor</td>
<td>Boeing Satellite Systems</td>
<td>Aalborg University</td>
<td>NRL</td>
</tr>
<tr>
<td>Country of Contractor</td>
<td>Multinational</td>
<td>Denmark</td>
<td>USA</td>
</tr>
<tr>
<td>Launch Site</td>
<td>Baikonur Cosmodrome</td>
<td>Satish Dhawan Space Center</td>
<td>Cape Canaveral</td>
</tr>
<tr>
<td>Launch Vehicle</td>
<td>Proton</td>
<td>PSLV</td>
<td>Delta 4 Heavy</td>
</tr>
<tr>
<td>Source Used for Orbital Data</td>
<td><a href="http://www.satellitedebris.net">www.satellitedebris.net</a> 12/12</td>
<td>SC - ASCR</td>
<td>SC - ASCR</td>
</tr>
<tr>
<td>Inclination radians</td>
<td>0.9005899</td>
<td>1.721418241</td>
<td>0</td>
</tr>
</tbody>
</table>

denoting both the true type of orbit and the type of orbit predicted by the random forest for each satellite in the training data. A monitoring model is then trained on the error pattern and the input features (see Table 1), excluding the target variable (Type of Orbit). Figure 2 shows how the random forest model and monitoring model were created and trained using BayesDB’s Meta-Modeling Language. Figure 3 shows how the monitoring model finds likely dependencies between the error pattern and input features.

One important function of the generative monitoring model is to predict error pattern given a set of input features. Figure 4 (a) and (b) show an example of using the monitoring model to predict error pattern for a particular satellite. Predicted error pattern can also be distilled into a more coarse-grained prediction of whether the black box discriminative model is expected to be correct or incorrect. If the true classification and predicted classification in the predicted error pattern are the same, then the monitoring model predicts that the black box model will classify the input correctly. This is regardless of the input’s true class. For example, if the monitoring model predicts an error pattern denoting that a satellite’s true type of orbit is polar and that the discriminative model will classify it as polar, then the monitoring model predicts that the black box model will be accurate in its prediction. This is the case even if the satellite’s true type of orbit is sun-synchronous; it is independent of the satellite’s true type of orbit. In this way, the monitoring model’s predictions for error pattern can be used as a metric for confidence in the black box model’s prediction for a given input. Figure 4 (c) shows the probability that the discriminative black box model is predicted to be correct or incorrect based on a particular satellite’s predicted error pattern.

In addition to being used to predict likely error patterns, the generative monitoring model can be used to simulate values of input features given a particular error pattern. Figure 5 shows the results of using the monitoring model to simulate purpose of satellites given a certain type of error pattern. Simulations from the monitoring model can also accept constraints on other input variables; in Figure 5, purpose is simulated both with and without an additional constraint.

Current research is focused on (i) empirically studying the behavior of this monitoring strategy on a broader problem class; (ii) developing analyses of the CrossCat monitoring model that enable qualitative and quantitative characterizations of ‘safety margins’ and overall reliability of the black-box method; and (iii) identifying use cases where interactive, ad-hoc querying of the monitor model can yield meaningful insights as judged by domain experts.
CREATE POPULATION satellites  
  FOR training_data WITH SCHEMA {  
    MODEL  
      Power_watts, Period_minutes, Apogee_km, Perigee_km,  
      Date_of_Launch, Eccentricity, Launch_Mass_kg  
    AS NUMERICAL;  
    MODEL  
      Country_of_Operator, Operator_Owner, Users, Purpose, Class_of_Orbit,  
      Type_of_Orbit, Contractor, Country_of_Contractor, Launch_Site,  
      Launch_Vehicle, Source_Used_for_Orbital_Data  
    AS NOMINAL;  
    MODEL Inclination_radians AS CYCLIC;  
    IGNORE Name };  

CREATE METAMODEL random_forest  
  FOR satellites WITH BASELINE crosscat(  
    OVERRIDE GENERATIVE MODEL FOR Type_of_Orbit  
    GIVEN Perigee_km, Apogee_km, Period_minutes  
    USING random_forest (k=6));  

INITIALIZE 25 MODELS FOR random_forest;  
ANALYZE random_forest FOR 1000 ITERATIONS;  

INSERT INTO monitor_data (classification_error_pattern)  
  (SELECT Type_of_Orbit  
   FROM satellites WHERE Name = "Iridium 60")  
|| " " ||  
(SIMULATE Type_of_Orbit  
   FROM random_forest WHERE Name = "Iridium 60" LIMIT 1;)

CREATE POPULATION monitor_population  
  FOR monitor_data WITH SCHEMA {  
    MODEL  
      classification_error_pattern, Power_watts, Period_minutes, Apogee_km,  
      Dry_Mass_kg, Perigee_km, Date_of_Launch, Eccentricity, Launch_Mass_kg  
    AS NUMERICAL;  
    MODEL  
      Country_of_Operator, Operator_Owner, Users, Purpose, Class_of_Orbit,  
      Contractor, Country_of_Contractor, Launch_Site, Launch_Vehicle,  
      Source_Used_for_Orbital_Data  
    AS NOMINAL;  
    MODEL Inclination_radians AS CYCLIC;  
    IGNORE Name };  

CREATE METAMODEL monitor_model  
  FOR monitor_population WITH BASELINE crosscat;  

INITIALIZE 25 MODELS FOR monitor_model;  
ANALYZE monitor_model FOR 1000 ITERATIONS;  

Figure 2: Predictive model and monitoring model with MML and BQL. The top box (a) shows the Meta-Modeling Language (MML) code used to create and train the random forest model. The middle box (b) shows Bayesian Query Language (BQL) code for creating and inserting error pattern into the monitor data for a particular satellite (Iridium 60). The bottom box (c) shows the MML code used to create and train the monitoring model.
Figure 3: Dependence of error pattern on input features. The top box (a) shows BQL code to create a heatmap of the pairwise probabilities of dependence between input features and the error pattern, the results of which are displayed in (b). The heatmap (b) shows how CrossCat hierarchically clusters the error pattern with other variables on which it is likely dependent. The input variables’ probabilities of dependence on error pattern are displayed in histogram (c). Variables in blue boxes in both (b) and (c) have the highest likelihood of dependence on the error pattern, i.e. \( P[I(x^d; \varepsilon) > 0] \approx 0.6 \).
Figure 4: Predicting error pattern. The top box (a) depicts BQL code to simulate error pattern for the satellite Iridium 60 using the generative monitoring model. The simulations of probable error pattern can be described as $\varepsilon \sim P_{\phi}(\varepsilon|\vec{x} = \vec{x}_s)$, where $\vec{x}_s$ denotes the set of input feature values for the satellite Iridium 60. The middle plot (b) illustrates the results of running the query: each error pattern is shown on the y-axis and its probability in a series of simulations is shown on the x-axis. In this case, the true error pattern for this satellite was "True: Intermediate, Predicted: Intermediate", highlighted in a green box. It was the second most likely error pattern predicted by the generative monitoring model, illustrating the monitoring model’s ability to produce reasonable predictions of error pattern given a set of input features. The bottom plot (c) divides error pattern $\varepsilon$ into two components: true class $\varepsilon_t$ and predicted class $\varepsilon_p$. It shows the probability that, based on the predicted error pattern for a satellite $\varepsilon$, the discriminative black box model was correct ($\varepsilon_t = \varepsilon_p$) or incorrect ($\varepsilon_t \neq \varepsilon_p$) in its prediction of orbit type for the satellite Iridium 60. Here, $\mathbb{I}[\cdot]$ denotes the indicator function. For comparison, the average true accuracy of the discriminative model was 37.5%.
Figure 5: **Simulating inputs conditioned on error pattern.** The probabilistic generative monitoring model allows for simulating input features given a particular error pattern. The top box (a) depicts BQL code for simulating purpose of a satellite given the error pattern "True type of orbit: polar, Predicted type of orbit: intermediate," and the second box (b) depicts BQL code for the same simulation with an additional input constraint setting users to military. The plot (c) shows side-by-side distributions of simulated purpose for satellites with both without any further constraints (red) and constraining the satellite’s users to be military (blue). Simulating purpose for this particular error pattern without additional constraints can be described as $x_i \sim P_{\phi}(x_i | \epsilon = \epsilon^*)$, where $x_i$ denotes purpose and $\epsilon^*$ is "True type of orbit: polar, Predicted type of orbit: Intermediate." With the additional input constraint, the simulations can be described as $x_i \sim P_{\phi}(x_i | \epsilon = \epsilon^*, x_j = x_j^*)$, where $x_j$ denotes users and $x_j^*$ is military.
References


