

Closed-Loop Uncertainty: Evaluation of Uncertainty as a Probability of Machine Correctness

ZACHARY BISHOF

CHRISTOPER MICHAEL

*Cognitive Geospatial Systems Section
Ocean Sciences Division*

November 13, 2023

DISTRIBUTION STATEMENT A: Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. **PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.**

1. REPORT DATE (DD-MM-YYYY) 13-11-2023			2. REPORT TYPE NRL Memorandum Report		3. DATES COVERED (From - To) 11-04-2021 – 10-04-2021	
4. TITLE AND SUBTITLE Closed-Loop Uncertainty: Evaluation of Uncertainty as a Probability of Machine Correctness					5a. CONTRACT NUMBER	
					5b. GRANT NUMBER	
					5c. PROGRAM ELEMENT NUMBER NISE	
6. AUTHOR(S) Zachary Bishof and Christopher Michael					5d. PROJECT NUMBER	
					5e. TASK NUMBER	
					5f. WORK UNIT NUMBER N20N	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Research Laboratory 4555 Overlook Avenue, SW Washington, DC 20375-5320					8. PERFORMING ORGANIZATION REPORT NUMBER NRL/7340/MR--2023/7	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Naval Research Laboratory 4555 Overlook Avenue, SW Washington, DC 20375-5320					10. SPONSOR / MONITOR'S ACRONYM(S) NRL-NISE	
					11. SPONSOR / MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION / AVAILABILITY STATEMENT DISTRIBUTION STATEMENT A: Approved for public release; distribution is unlimited.						
13. SUPPLEMENTARY NOTES Karles Fellowship						
14. ABSTRACT Uncertainty values reported by machine learning models do not often accurately predict the probability of model correctness. Especially for interactive machine learning applications where curated labeled data may be scarce and data streams change modality, the conventional use of data-model statistics often fall short. This may be due in large part to the absence of feedback of correctness, which is feasible for a large number of human-in-the-loop applications. In this study, we explore the potential of incorporating such feedback into an interactive machine learning model for a threshold selection problem. The problem involves a user subjectively selecting the point at which they consider a signal to transition to a low state without providing specific rules to the user for what constitutes a low state. The classifier-driven machine model will attempt to mimic the analyst's selection, with the signal becoming more complex in time. A methodology for evaluating uncertainty, which we define as the probability of machine correctness, is presented and used to compare a baseline model using naïve Bayes with a novel reinforcement learning approach. The novel approach refines a black-box model for uncertainty by incorporating machine performance as feedback. Experiments are conducted over a large number of realizations in order to properly evaluate uncertainty using a stochastic process. Results show that our novel approach, called closed-loop uncertainty (CLU), outperforms the baseline in every case, yielding about 46% improvement over the baseline on average.						
15. SUBJECT TERMS						
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON Zachary Bishof	
a. REPORT U	b. ABSTRACT U	c. THIS PAGE U			U	17

This page intentionally left blank.

1. Abstract

Uncertainty values reported by machine learning models do not often accurately predict the probability of model correctness. Especially for interactive machine learning applications where curated labeled data may be scarce and data streams change modality, the conventional use of data-model statistics often fall short. This may be due in large part to the absence of feedback of correctness, which is feasible for a large number of human-in-the-loop applications. In this study, we explore the potential of incorporating such feedback into an interactive machine learning model for a threshold selection problem. The problem involves a user subjectively selecting the point at which they consider a signal to transition to a low state without providing specific rules to the user for what constitutes a low state. The classifier-driven machine model will attempt to mimic the analyst’s selection, with the signal becoming more complex in time.

A methodology for evaluating uncertainty, which we define as the probability of machine correctness, is presented and used to compare a baseline model using naïve Bayes with a novel reinforcement learning approach. The novel approach refines a black-box model for uncertainty by incorporating machine performance as feedback. Experiments are conducted over a large number of realizations in order to properly evaluate uncertainty using a stochastic process. Results show that our novel approach, called closed-loop uncertainty (CLU), outperforms the baseline in every case, yielding about 46% improvement over the baseline on average.

In addition we evaluate a Kernel Density Estimation naïve Bayes model, an induction model, and a constant model as the black-box model to test the performance of CLU.

2. Introduction

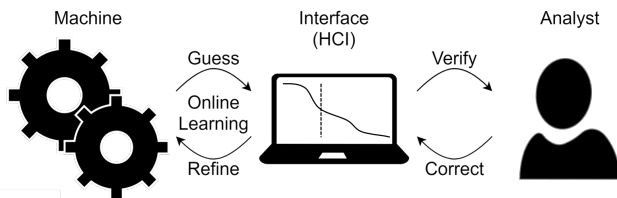


Figure 1. The interactive machine learning paradigm tightly couples machine learning to a human analyst via an intuitive interface for a task.

The concept of uncertainty is used with great frequency in machine learning (ML) to give an understanding of the

dependability of model classifications and predictions. However, uncertainty is interpreted and used in many different ways when applying ML models. It is used to evaluate the reliability of the ML (Abdar et al., 2021; Jiang et al., 2018), to optimize ML, (Sahinidis, 2004), and to provide transparency to stakeholders about the ML (Bhatt et al., 2021). In active learning, uncertainty reduction is a vital component where uncertainty establishes a basis in deciding what examples to query the user for labeling to maximize the precision in a data stream (Aggarwal et al., 2014; Hüllermeier and Waegeman, 2021). Interactive machine learning (IML) involves tightly coupled ongoing interactions between an ML algorithm and a human via a constrained human-computer interface (Mosqueira-Rey et al., 2022). IML implementations utilize a human-machine team that cooperate to iteratively solve a problem. As such, it is useful to define uncertainty as the probability that a machine’s attempt to solve a problem is incorrect (Monarch, 2021; Michael et al., 2019). When defined in this way, uncertainty can be used to manage cognitive load on the user by maintaining balance between exploration and exploitation. In theory, ML models that have a statistically viable sampling of data may yield accurate values of uncertainty solely based on the distribution of this sampling under a robust data model. However, many problems either do not attain this sampling or suffer from *concept drift*, a modality change in data context that interrupts or invalidates the data model (Schlimmer and Granger, 1986). This modality change makes it likely that the data model’s yielded uncertainty is low while classification accuracy is also low, which is indicative of an uncertainty model that does not properly reflect the probability of machine correctness. The opposite may also be true, where a yielded high uncertainty is reported during events of high accuracy. This tends to occur during classifier warm-up or when data models are underfit.

A major advantage of the IML paradigm is the presence of a human user who may iteratively and immediately correct machine error. In some cases, this human feedback may be available immediately to refine a supervised data model by training online on user-corrected information treated as ground truth. Though this technique is trivial for the precision of the ML model’s classification or prediction, very little work has focused on incorporating such feedback to improve the way in which uncertainty is quantified (Michael et al., 2020; Monarch, 2021).

The focus of the research presented in this paper is to better understand how uncertainty is improved and evaluated for IML applications. A methodology for experimentation that takes iterative feedback of machine correctness into account is presented. To introduce the underlying concepts, a somewhat subjective and generative thresholding task is defined and used as a target problem for experimentation. This task involves a human analyst selecting the point at which a de-

caying signal, namely a sigmoid, is to be considered “low” on a visual plot. As the task progresses, the signals enter new stages of modality to simulate concept drift. At every step, the machine’s goal is to place the threshold within an accepted tolerance to the analyst’s placement. The machine begins the task at *cold start*, meaning with no prior training data, and trains on the human placement using a supervised model. Additionally, and most importantly for our case, the machine will also provide a probability that this placement is correct.

The presented methodology for evaluation examines the machine’s reported uncertainty over many independent realizations of the task in order to compare it to a more accurate measurement of the probability of correctness. Statistics are gathered at every step across all realizations to evaluate the performance of the uncertainty model. We present and compare two supervised models for uncertainty using the presented methodology: A baseline that implements a conventional data-model approach using naïve Bayes and a novel approach using reinforcement learning to adjust the bias of the baseline using feedback of machine correctness. We name the novel approach the *closed-loop uncertainty* (CLU) model because it takes into account machine correctness in an online and iterative manner.

3. Related Work

The discussion and formulation of uncertainty is an extensive topic in ML literature. The majority of prior work discussed in this section mainly focuses on studies that present methods for calculating some value of uncertainty based on the distribution of example data. Studies involving uncertainty values that are provided explicitly by humans for example data during training are considered out of the scope of this study. We do not know of any study that feeds back correctness to improve uncertainty modeling as a probability of machine correctness.

The interpretation of confidence, which we define to be the opposite of uncertainty, by Pronk et al. (2005) for the naïve Bayes classifier formulates the confidence interval for the posterior probabilities of classes. The CLU model we present contrasts to this approach in that it takes posterior probabilities as an observable state rather than an estimate for classification confidence.

Defining uncertainty as the probability that a model is incorrect (or confidence as the probability that a model is correct) is useful for evaluating trust in a model or metering the cognitive load and human interaction. However, this definition does have limitations when compared to other discussions in literature. One limitation is that it fails to distinguish between *aleatoric* uncertainty and *epistemic* uncertainty (Hüllermeier and Waegeman, 2021; Hora, 1996).

Aleatoric uncertainty involves the distribution of noise and other randomness within the data, while epistemic uncertainty addresses the lack of knowledge within the ML model. Aleatoric uncertainty is difficult to measure (Wang et al., 2019), especially under concept drift (Lu et al., 2020). Other studies more aligned with our approach have defined uncertainty as a measurement of what is not known at the time of classification (Klās and Vollmer, 2018). Though this definition allows considerably more leeway, we choose a probabilistic interpretation that allows us to validate models for uncertainty experimentally within an IML paradigm.

Though much of ML theory models classification and prediction on the basis of statistical probability, general formulation and evaluation of uncertainty is known to be considered something of an afterthought (Klās and Vollmer, 2018). The general idea behind most models of uncertainty is a mathematical basis for the data model. For example, the softmax layer in a neural network gives a score that may indicate a lack of knowledge about a classification for multi-classification problems (Jiang et al., 2018). However, many times such models demonstrate some sort of best fit to the probabilities of a label in training data rather than a logical measurement of machine knowledge about a specific class (Kaplan et al., 2018). Additionally, these models are often not well calibrated or may not adequately reflect the probability of the machine to be incorrect after calibration (Guo et al., 2017). In these cases, a classification with a low uncertainty does not necessarily imply a high accuracy (Provost et al., 1998). For streaming problems, these shortcomings of conventional ML models for uncertainty are exacerbated in the presence of concept drift where data modalities in the stream may change abruptly and/or unexpectedly (Lu et al., 2020). Therefore, it may be worthwhile to consider if an interpretation of uncertainty that departs from stringent mathematical definitions for the data model would be practical in providing a more accurate quantification of uncertainty, especially in situations where sampling is low or concept drift is expected to occur.

Mathematical frameworks for estimating uncertainty have been explored in the context of neural networks for image processing (Wang et al., 2019). These techniques use variance and entropy of a statistical distribution to measure uncertainty, which could also aid in detecting concept drift (Du et al., 2014). Our CLU model differs from these techniques in that it does not explicitly detect concept drift or variance in input feature data. Rather, CLU only observes some measurement of uncertainty, namely the posterior probabilities of a black-box classifier, and adjusts its bias based on the observed accuracy of the machine. Therefore, the goal of our work is not to detect concept drift, but to provide a model of uncertainty that is adaptable to concept drift when the drift causes bias in the underlying model.

The CLU model presented in this study uses a feedback model to yield an improved quantification of uncertainty, and this value may have a higher order uncertainty associated with it. Such phenomena for reinforcement learning have been discussed thoroughly in [Clements et al. \(2020\)](#), where the difference between aleatoric and epistemic uncertainty is distinguished within deep reinforcement learning. The study builds on previous work that aims to view uncertainty through the lens of a return distribution and the variance associated with it ([Nikolov et al., 2018](#); [Bellemare et al., 2017](#)). These previous studies in reinforcement learning have defined uncertainty as a function of the input data or lack of input data, while the CLU implementation that we present defines uncertainty as a function of the accuracy of the underlying classification model within a stochastic process where accuracies may be measured.

IML models have calculated uncertainty using mixture models or some approach that resembles that of the machine learning algorithm used in the classification process ([Jiang et al., 2019](#)). In other studies, uncertainty has been defined as the sample conditional error, which is the probability that the classifier makes a mistake on a given sample ([Li and Sethi, 2006](#); [Teso and Vergari, 2022](#)). These techniques require that the underlying distributions of the data model are known, which is often not possible. Our approach allows for a black-box baseline model of uncertainty, and we show that it performs accurately even when the data-model distributions are not accurate.

4. Background

The notation $[[a, b]]$ is used to signify the set of integers from a to b , that is $[[a, b]] = \{a, a + 1, \dots, b - 1, b\}$ where $a, b \in \mathbb{I}$; while the traditional notation $[a, b]$ refers to the continuous inclusive set of real numbers from a to b .

A sigmoid curve, or logistic curve, is an S-shaped curve. For our setting, we are interested in a version that begins at near $y = 1$ and decays to approach $y = 0$, as well as being centered somewhere between $x = 0.2$ and $x = 0.8$. In general, we can describe this curve using the following function:

$$y(x) = \frac{1}{1 + e^{k(x-x_0)}} \quad (1)$$

where x_0 is the curves midpoint and k is the decay rate.

Gaussian naive Bayes classification is used as a baseline, which operates by calculating posterior probabilities assuming all predictor covariates have a normal distribution and are independent. ([Zhang, 2004](#))

Kernel Density Estimation (KDE) can be used to estimate the distribution of a covariate. It is useful in the sense that it

can help deal with the problems posed by continuous variables. If some of the features can be described as continuous and some can be described as discrete, then assuming they all come from the same distribution is not valid. KDE allows for the smoothing of the distributions of the feature space. ([John and Langley, 2013](#))

A Markov Decision Process (MDP) is defined as a tuple (S, A, T, R, γ) , which contains a state space S , a set of possible actions A , a transition probability matrix T , a reward function R , and a discounting factor γ . ([Sutton and Barto, 1998](#))

Q-Learning is a type of model-free RL that can be used to find the optimal policy, the strategy for choosing actions in each state, by updating the expected reward for state-action pairs. The only conditions for convergence on the optimal policy is that the actions are repeatedly sampled in all states, and the action-values are discrete ([Watkins and Dayan, 1992](#)). This is advantageous for our problem, because it allows us to approach the optimal policy simply by randomly choosing actions for each trial. Q-Learning is a popular implementation of RL due to its simplicity and effectiveness at finding optimal solutions in MDPs.

In experience replay, instead of updating the expected reward for state-action pairs as they appear in simulation, there are stored data. There is a set of past experiences which depict the states, actions, rewards, and next states. This means that the learning is separate from the experience, and all interactions with the environment are stored.

4.1. Index of Terms

Uncertainty - A measure of what we don't know about the classification. In a practical sense, the probability that the classifier algorithm makes an incorrect classification.

Confidence - The opposite of uncertainty. $Confidence = 1 - Uncertainty$

Accuracy - The fraction of correct classifications over total classifications.

Sample Points - 100 points on a trial graph, evenly spaced along the x-axis, which together depict a sigmoid curve.

Trial - An individual graph where the analyst has made a selection of the correct location of a human placed threshold. Also included is all of the data that comes from this graph and selection.

Phase - A round of seven trials for which the concept drift is held conceptually fixed. For example, in the first phase only square waves are displayed, the second

phase only displays logistic curves, and the third phase shows random noise at random selected points.

Realization - A complete run of the experiment, which contains 35 trials and 5 total phases. In total 30 realizations were performed.

Baseline Uncertainty Model - The model that measures uncertainty using conventional approaches rooted in naïve Bayes. This is the model that is built upon by feeding into CLU. We can think of this as the black-box displayed in figure 4

Performance - How well the confidence values from the uncertainty model matches the probability of correctness, which can be measured with the distance between the accuracy and the average confidence values produced.

Human Placed Threshold - The location of the threshold on the graph that the analyst chooses to be where the sigmoid curve signal is no longer considered high.

Machine Placed Threshold - The location that the classifier algorithm predicts is the location of the human placed threshold.

Tolerance (T) - A parameter representing the maximum distance, with respect to the horizontal, that a machine placed threshold can be from the human placed threshold for the classification to be considered correct.

Optimal Policy - At any given step or trial in the realization, the policy that chooses the action associated with the maximum available action value function.

Present Policy - A modification on the optimal policy at a specific step or trial in the realization. The state associated with the trial is given a random action, and this state action pair is swapped into the optimal policy. All other state action pairs stay the same.

5. Threshold Selection Problem

The methodology for uncertainty presented in this study is specific to an online IML implementation (Fails and Olsen Jr, 2003). Though a multitude of problems for which IML implementations exist are available in the current state of the art, the problems either possess overly complex features and interfaces or do not provide controls to induce concept drift in a stochastic manner. We present a threshold selection problem that exhibits the properties of being an intuitive task definition with a simple interface, a 2-dimensional dataset with a minimal feature space, a stochastic basis for generating a very large number of 2D examples for the problem space, moderate subjectivity that prevents trivial

solutions without human interaction, and a parameterized complexity and noise model used to induce concept drift.

This threshold selection problem is a simpler surrogate for online IML applications in the sense that it exhibits subjectivity in preference and concept drift, similar to those discussed by Kabra et al. (2013) and Michael et al. (2019). This is all while being simpler to form into a stochastic process by which we can study uncertainty as a probability of correctness.

The problem is able to be realized such that a single trial of the problem will be, for the most part, similar in complexity across all realizations. This property is especially useful for the methodology of evaluating uncertainty as a probability of correctness.

5.1. Trial Definition

The decaying sigmoid curve used for generating examples is a type of logistic function that begins at near $y = 1$, decays to approach $y = 0$, and is centered somewhere between $x = 0.2$ and $x = 0.8$. In general, the curve may be described using the following function:

$$y(x) = \frac{1}{1 + e^{k(x-x_0)}} \quad (2)$$

where $x_0 \in [0.2, 0.8]$ is the curve's midpoint and k is the decay rate.

A *trial* consists of a 2D plot of a decaying sigmoid curve sampled at 100 regular discrete points. The user is asked to locate the point at which they think the sigmoid transitions from a high state to a low state, i.e. the human-placed threshold, which is treated as ground truth.

5.2. Phase Definition

In a realization of the problem, multiple trials will be generated and presented to the human-machine team in a particular order. A *phase* is a consecutive subset of trials with similar stochastic parameterization, which defines a set modality of complexity. Overall, the progression of phases are intended to induce some new form of complexity to the sigmoid.

Phase I will only contain trials that are square waves. This phase is the simplest phase and the only phase that leaves little room for subjectivity. The only stochastic parameter is the center of the plot, $x_0 \in [0.2, 0.8]$, which is selected in a uniform random way. The curve depicted for the analyst in each trial for phase I is

$$y(x) = \begin{cases} 1 & \text{if } x \leq x_0 \\ 0 & \text{if } x > x_0 \end{cases} \quad (3)$$

Phase II depicts a logistic curve where the decay rate is

randomly assigned, leading to a faster or slower decay:

$$k = \left(1 - \frac{1}{1 + \exp\left(\frac{-1}{1-d}\right)} \right)^{-1} \quad (4)$$

where $d \in [0, 1)$ is a uniform random variable. This formula was found to yield the best mix of sigmoid decay after experimenting with a variety of other approaches.

The subsequent phases, III-V, induce random noise throughout the sigmoid. The noise N is applied in an additive way. The magnitude of the noise $N = m \cos(T\pi)$, where $T \in [0, 1]$ and $m \in [0, 0.8]$ are uniform random variables. The variable m indicates the maximum magnitude of additive noise such that $N \in [-m, m]$. Trials of Phase III generate sigmoids with N added to a uniformly random percentage of uniformly random chosen points. Phase IV generates sigmoids with N added to randomly chosen subintervals of points. Subinterval sizes are chosen by the uniform random variable $l \in [0, 100]$. The number of number of subintervals is randomly chosen from the set $\{0, 1, \dots, \lfloor \frac{100}{l} \rfloor\}$. Finally, Phase V sigmoids contain additive noise, N , throughout the entire function. Any phase may theoretically generate a plot from a previous phase depending on the parameters and noise.

An example of a trial from each phase is shown by the plot in Figure 2. The examples shows both the human-placed threshold in blue, which is considered ground truth, and a machine-placed threshold in red, which is placed algorithmically based on a naïve Bayes classifier. As will be discussed in Section 6.1, the methodology allows for different tolerances for machine-placed thresholds to be considered as correct placements.

6. Machine Models

6.1. Naïve Bayes Classifier

A Gaussian naïve Bayes classifier is used to predict the location of the human placed threshold. We refer to this predicted threshold as the *machine-placed threshold*, which is shown as the red lines of the example plots in Figure 2. The features used by the naïve Bayes classifier include the basic coordinates of each trial’s sigmoid. In addition, several features are extracted to enhance the feature space. This includes the first derivative of the sigmoid at each discrete point, the second derivative of the sigmoid at each point, the mean of the next 10 point’s y values, and the mean of each point along with its 2 direct neighbors. These features were determined using conventional feature selection techniques, and they were shown to yield generally high accuracy in each phase after warm up for reasonable tolerances of machine placement.

Labels are determined by assigning 1 to all sample points

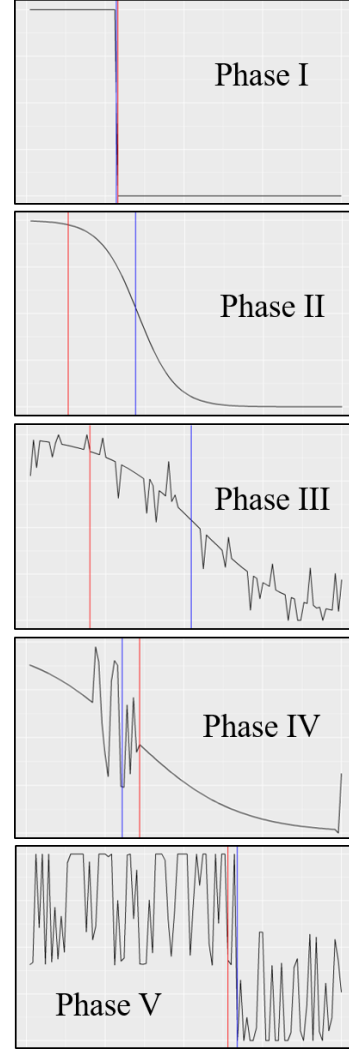


Figure 2. An example trial of each phase showing the human-placed threshold in blue and the machine-placed threshold in red. Correct machine placement is determined by the tolerance, where a tolerance of 0.04 would mark the Phase I and V examples correct and all others incorrect. A tolerance of 0.06 would allow for the Phase IV example to be marked for correct machine placement. Human-placed thresholds are somewhat subjective due to the preference of the analyst.

with x-coordinates less than or equal to the human placed threshold and 0 to all sample points with x-coordinates greater than the human placed threshold. This labeling scheme worked best for machine placement when compared to other schemes, mainly due to its relatively balanced positive and negative labels when training.

The location of the machine placement was chosen at the first point with the mean of the posterior probability (generated from Gaussian naïve Bayes) and that of its two direct neighbors being greater than 0.55.

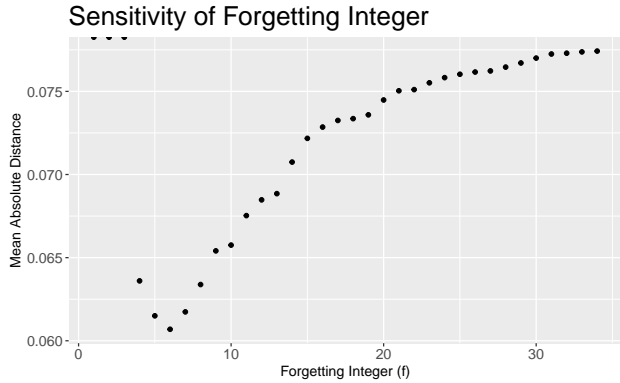


Figure 3. The forgetting integer (f) which results in the smallest average distance between the machine placement and the human placement is $f = 6$

When analyzing the performance of naïve Bayes, it was found that the retention of all training instances hurt performance in later phases due to bias. The most optimal classifier only considered the instances of the 6 preceding trials when training, as seen in figure 3. This type of forgetting is typically known to be beneficial for drifting data streams (Krawczyk and Wozniak, 2015), and thus was used for our experimentation.

6.2. Baseline Uncertainty Model

A baseline uncertainty model is used to produce first-order input uncertainty values for CLU. Primarily the focus of this research was to utilize a more conventional uncertainty model based very heavily on the naïve Bayes classifier of the previous section and is used for experimentation.

This is referred to as a *baseline uncertainty model* mainly because it resembles a natural choice for an uncertainty model in the current state of the art; that is, one that does not iteratively take feedback of correctness into account. Unlike the model for machine placement, the uncertainty model must account for some *placement tolerance*, denoted by the variable T , allowed by the application. The placement tolerance is an independent variable that determines the distance within which a machine placement must be from a human placement to be considered correct. The lower the placement tolerance, the lower the expected placement accuracy, and vice versa.

In labeling a trial for the uncertainty model, all sample points within the distance of the placement tolerance are labeled as 1, and all other points as 0. A Gaussian naïve Bayes classifier is trained with this information. Using this model, the posterior probability that a sample point in a trial should be given a label of 1 may be calculated. A confidence (recall $confidence = 1 - uncertainty$) can

then be generated from this value by taking the average of all these probabilities within the placement tolerance of the machine-placed threshold.

Other than labeling, the baseline uncertainty model differs from the naïve Bayes classifier in two other ways. First, we found during experimentation that instance forgetting does not yield a more accurate value of uncertainty, so the baseline model does not implement forgetting. The second difference is that the baseline’s technique for labeling generates a very large amount of bias for certain features, namely the first and second derivative features, which was found through feature selection analysis. Therefore, the baseline uncertainty model uses the same input feature set as the naïve Bayes classifier except for the first and second derivative features.

6.3. Closed-Loop Uncertainty

We define the Markov Decision Process and the policy selection. The size of the state space, action space, and the window parameter in the reward function were studied using a sensitivity analysis described in section 6.5.

6.3.1. STATE SPACE

We define the state space $S = \{1, 2, 3\}$ by taking a discretization of the difference between successive confidence values

$$\Delta C_t = C_t - C_{t-1} \quad (5)$$

where t indicates the trial number, generated by the baseline model within any realization. This is done by dividing the interval $[-1, 1]$ into three equal sized sets. For example, if trial number 7 gives a confidence of 0.9, and trial number 8 gives a confidence of 0.6, then ΔC_8 would be -0.3, and trial number 8 would be in state 2. We do a discretization in this way, because the state space is required to be discrete to guarantee convergence on the optimal policy (described in section 6.3.6). Conventional discretization techniques resemble this approach (Hasselt, 2012).

$$S_t(\Delta C_t) = \begin{cases} 1 & \text{if } \Delta C_t \in [-1, -0.\overline{33}) \\ 2 & \text{if } \Delta C_t \in [-0.\overline{33}, 0.\overline{33}) \\ 3 & \text{if } \Delta C_t \in [0.\overline{33}, 1] \end{cases} \quad (6)$$

6.3.2. ACTION SPACE

The set of possible actions

$$A = \{-1, -0.95, 0.90, \dots, 0.95, 1\} \quad (7)$$

is defined by evenly discretizing the interval $[-1, 1]$ into 41 values.

This definition allows for a fractional shift on any given state’s confidence value either up or down. For example,

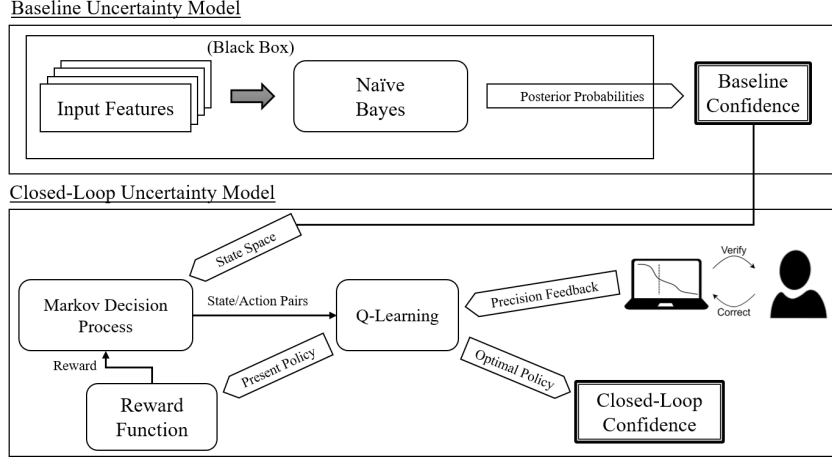


Figure 4. Architecture of Closed-Loop Uncertainty

an action of 0.5 can be thought as the act of making a 50% gain of any given state’s confidence value towards 1, while the action -0.5 would cause a 50% loss towards zero .

More formally, given an action A_t and a confidence value C_t , we can find the confidence value that is the result of an action $C'_t(A_t, C_t)$, by using the following equation:

$$C'_t(A_t, C_t) = \begin{cases} C_t + A_t C_t & \text{if } A \leq 0 \\ C_t + A_t(1 - C_t) & \text{if } A > 0 \end{cases} \quad (8)$$

6.3.3. STATE TRANSITIONS

Previously we defined ΔC_t as the difference between successive baseline confidence values, and this was used to define the state space. We now must define S'_t , that is the state that is transitioned to from S_t as a result of action A_t . Let

$$\Delta C'_t = C'_t - C_{t-1} \quad (9)$$

It should be noted that $\Delta C'_t \neq C'_t - C'_{t-1}$. S'_t reflects the state that occurs under the change to the bias from action A_t . For this reason, we replace the first term from equation 5 with the result from equation 8

The new state S'_t , not to be confused with S_{t+1} , is calculated using equation 6, and thus becomes $S'_t = S(\Delta C'_t)$.

6.3.4. TRANSITION PROBABILITIES

The transition probabilities T are not needed to be known, as we are dealing with model-free reinforcement learning. (Sutton and Barto, 1998)

6.3.5. REWARD FUNCTION

For CLU, an ideal reward function would give a higher reward when the policy is choosing actions that tune the

bias of the confidence in the direction of the probability of a correct classification. For this reason, the reward function uses an estimate of the error between the confidence of the baseline model and that of the accuracy of the machine placement.

Define accuracy p_t at trial t to be equal to either 1 when the machine placement is correct, i.e. when the machine placement is within a tolerance of the human placement, or 0 when the machine placement is incorrect. The mean streaming accuracy is defined as

$$\bar{p}_t = \min(w, t)^{-1} \sum_{i=\max(1, t-w-1)}^{t-1} p_i \quad (10)$$

Where the window size w is set equal to 3 (see 6.5.1). Note that \bar{p}_t is only used (and defined) for $t > 1$. Given a present policy π_t (which will be defined in equation 15) at trial t , a state S_t , and a window size w , the window mean squared error, which estimates the discrepancy between machine placement accuracy and baseline confidence values, is defined as

$$D'_t = \min(w, t-1)^{-1} \sum_{i=\max(1, t-w-1)}^{t-1} (C'_i(\pi_i(S_i), C_i) - \bar{p}_t)^2 \quad (11)$$

For the baseline confidences, i.e. the policy $\pi(s) = 0$ for all s , meaning that $C'_t = C_t$ at every t , the baseline window mean squared error D_t is

$$D_t = \min(w, t)^{-1} \sum_{i=\max(1, t-w-1)}^{t-1} (C_i - \bar{p}_t)^2 \quad (12)$$

These are used to calculate R_t for transitioning from state S_t to S'_t due to action A_t .

$$R_t = \begin{cases} 1 - D'_t & D'_t < D_t \\ -D'_t & D'_t \geq D_t \end{cases} \quad (13)$$

Note that at each trial t , D'_t depends on not only the mean streaming precision \bar{p}_t , but the present policy at each trial. This present policy is used to define C'_t for all trials in a given window W_t , i.e. trials $t \in W_t = [[\max(1, t - w - 1), t - 1]]$. These values for C'_t inside each window depend on the states and baseline confidences for each trial inside W_t , yielding a unique R_t .

6.3.6. POLICY SELECTION

At any trial, all of the previous trials' states and black-box output confidences are stored and actions are randomly assigned to each trial. This forms a basis for the online Q-learning model. The action-value function $Q(S, A)$ gives the expected reward for each state under each action.

The learning is defined in the typical way for Q-learning (see 6.4) where the discount factor and learning rate are both set to 0.1.

Given a state s , the optimal policy π_* yields the action that produces the highest q-value. If all q-values are either negative or zero, then the optimal policy is to take action "0".

$$\pi_*(s) = \begin{cases} \arg \max_a Q(s, a) & \exists a \in A \text{ s.t. } Q(s, a) > 0 \\ 0 & \text{otherwise} \end{cases} \quad (14)$$

Given a state S_t and randomly chosen action A_t at trial t , the present policy, π_t , replaces the optimal action $\pi_*(S_t)$ with A_t :

$$\pi_t(s) = \begin{cases} A_t & s = S_t \\ \pi_*(s) & s \neq S_t \end{cases} \quad (15)$$

The present policy was used to determine reward values.

The optimal policy at each trial is used to calculate the confidence values as shown in Figure 4.

6.3.7. EXPERIENCE REPLAY

Experiences $e_t = (S_t, A_t, R_t, S_{t+1})$ were stored in a dataset, which are then replayed to the agent. This technique allowed us to conduct and store the data from several realizations. This permitted us to conduct all of our model building after conducting the 30 realizations, because it allowed us to compare a wide variety of definitions of the MDP and evaluate results by trial across realizations.

6.4. The Choice to Use Q-Learning

As outlined in section 6.3.7, it was necessary to use experience replay in our experiment. As a result, it is necessary

to learn using off-policy techniques. As a result, Q-learning was a logical choice, because Q-learning operates off-policy (Mnih et al., 2013). This choice was made as opposed to something like SARSA where actions are chosen based off the current policy.

The learning rule for Q-learning is defined at each trial t as

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \lambda \left(R_t + \gamma \max_a Q(S'_t, a) - Q(S_t, A_t) \right) \quad (16)$$

where $\gamma, \lambda \in [0, 1]$ are the discount factor and learning rate respectively. We had both of these set to 0.1.

6.5. Sensitivity Analysis

Throughout the process of defining the MDP, several decisions had to be made regarding the value of key parameters. There was the window size w for the reward function presented in section 6.3.5, the size of the state space (which determines the size of the intervals in equation 6), and the number of available actions (which determines the members of the set in equation 7).

In order to make these selections, sensitivity analyses were conducted. The approach, for each of these parameters, was to hold all other parameters fixed and to run the experiment for a large set of the parameter in question. Then, results were collected using equation 19. The parameter value with the highest $1 - MAE$, across all trials and realizations, was then chosen.

6.5.1. WINDOW SIZE FOR REWARD FUNCTION

The window size w was the first parameter that was cycled through, with all other parameters held fixed. Arbitrarily, the size of the state space was held fixed at 3 and the number of possible actions was held fixed at 9. In all, the values tested were $w \in [[2, 25]]$.

We can see from figure 5 that the window size that resulted in the best performance was $w = 3$. That being said, there was not a major difference between highest and lowest performing window sizes (note the y-axis in figure 5). This leads us to conclude that, although the window size is important, small changes in the window size do not result in large changes in the overall performance of CLU.

6.5.2. SIZE OF THE STATE SPACE

As the state space size was tuned, the intervals that, if containing ΔC_t , determine S_t must cover $[-1, 1]$. Therefore the

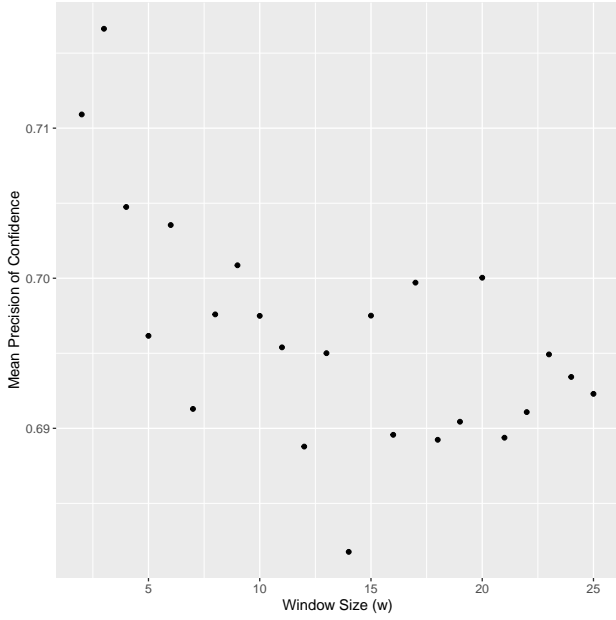


Figure 5. A window size of $w = 3$ is demonstrated to be the highest performing. Note that there is not a wide range of performances, thus leading us to conclude that CLU is not highly sensitive to the choice of w .

general definition of the state space is

$$S_t(\Delta C_t) = \begin{cases} 1 & \text{if } \Delta C_t \in [-1, -1 + \frac{2}{\sigma}) \\ 2 & \text{if } \Delta C_t \in [-1 + \frac{2}{\sigma}, -1 + \frac{4}{\sigma}) \\ \vdots & \\ \sigma & \text{if } \Delta C_t \in [-1 + \frac{2(\sigma-1)}{\sigma}, 1] \end{cases} \quad (17)$$

where $\sigma \in \mathbb{N}$ is the number of states. In figure 6, a smaller state space is shown to out perform a larger state space, with a peak performance at size 3. One of the causes for this might be that a larger state space needs more time to explore, and would never be fully utilized in the limited 35 trials for each realization. Recall that one of the conditions for the convergence of Q-learning is that all actions are repeatedly sampled in all states (Watkins and Dayan, 1992).

A caveat to this general principle of a smaller state space working better is that a state space of size 3 seemed to out perform a state space of size 2. A state space of size 2 sets all negative ΔC_t in state 1 and all positive ΔC_t in state 2. The downside to this definition is that ΔC_t that is near 0 could end up in either state. This means that for any value of ΔC_t that is near zero, which was quite common, a small change would result in a potentially large change in the policy. As a result, a state space of size 3 works better, because now the three possible states can be thought of as !: a significant negative decrease in successive confidence values, 2: no

significant change in successive confidence values, and 3: a significant increase in successive confidence values.

It can also be deduced from figure 6 that CLU is sensitive to the size of the state space. This means that any implementations of CLU should take great care in the choice for the state space.

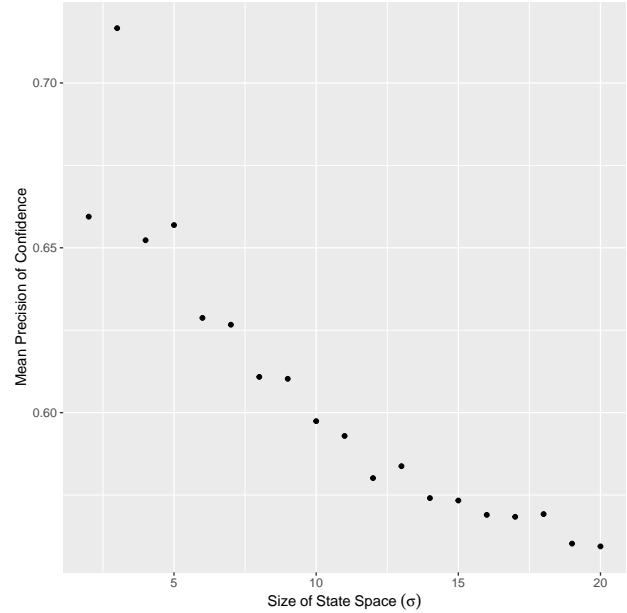


Figure 6. A state space of size 3 performs best, with all larger state spaces performing noticeably worse.

6.5.3. ACTION SPACE SIZE

As the action space was altered, the set of possible actions is

$$A = \{-1, \frac{3-\alpha}{\alpha-1}, \frac{5-\alpha}{\alpha-1}, \dots, 1\} \quad (18)$$

Where $\alpha \in 2\mathbb{N} + 1$ is the number of possible actions. α was required to be an odd number greater than or equal to 3, so that an action of 0 (i.e. the action that does nothing) was always possible, and every positive action had an equivalent negative action.

In figure 7, it appears that the choice of α does not seem to alter the performance of CLU. Note that, for $\alpha > 3$, the worst performing α and the best performing α do not result in much difference in overall performance. This tells us that CLU is not highly sensitive to the size of the action space.

In figure 7, there is an initial noisy period and then around $\alpha = 23$, there appears to be convergence. As such, we choose the highest performing action space size to be $\alpha = 41$.

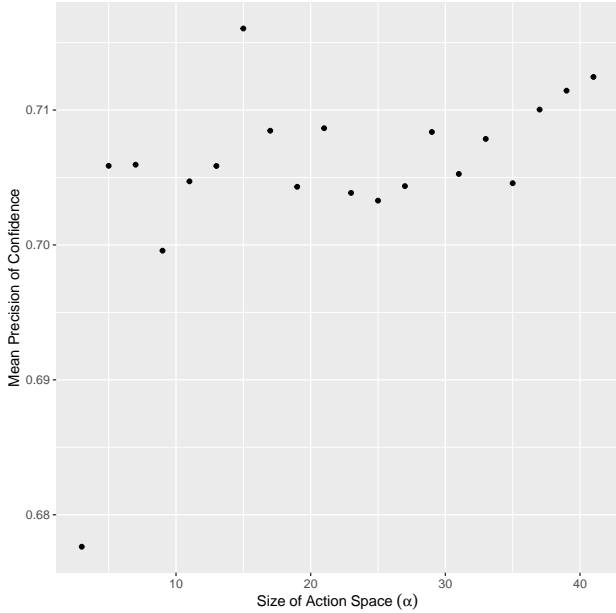


Figure 7. The performance of CLU with different action spaces.

7. Methodology and Experimentation

In order to evaluate uncertainty as a probability of incorrectness, the methodology for experimentation must be formed as a stochastic process. Judging an uncertainty model based on a single realization of an experiment is likened to evaluating the fairness of a coin based on a single flip. Therefore, the presented experimental methodology for evaluating uncertainty is based on many parallel realizations of the experiment that are evaluated at every trial t . Preferences from user to user are not expected to be similar for phases other than Phase I, but each user’s preference is expected to be consistent within a realization.

Results are reported using one minus the mean absolute error MAE at each trial t across all realizations:

$$PoC(t) = 1 - MAE(t) = 1 - \frac{1}{n(\mathbf{C}_t)} \sum_{c_i \in \mathbf{C}_t} |P_t - c_i| a \quad (19)$$

where \mathbf{C}_t is a set containing confidences at trial t across all realizations, $n(\mathbf{C}_t)$ represents the number of realizations in \mathbf{C}_t , and P_t is the accuracy of machine placement at trial t across all realizations. The subtractions from 1 is done so as to have higher values represent a better performing model. Sometimes we refer to $1 - MAE$ as *precision of confidence* or PoC .

Experiments were conducted for 30 realizations, each realization consisting of the 5 phases described in Section 5.2, each with 7 trials for a total of 35 trials per realization. This number of realizations was found to reach a statistically sig-

nificant sample size during experimentation. Ground truth was labeled for each trial of each realization by a user who was asked to maintain preference for the entire realization. All machine models began each realization with a cold start, meaning that all classifiers and uncertainty models begin the first trial of each realization with no training data. As discussed in Section 6.1, the classifier used for machine placement disregards all training prior to the 6 most recent trials, as this improved the accuracy of machine placement. The baseline uncertainty model did not implement this type of forgetting.

8. Results

Figure 8 shows the average results across all realizations for tolerances 0.02, 0.1, and 0.2 individually. The plots include the machine placement accuracy for reference. These particular plots are chosen as they compare and contrast the models for situations where machine placement accuracy is expected to be low ($T = 0.02$), moderate ($T = 0.1$), and high ($T = 0.2$).

The baseline model (the red line) dramatically underperforms during phase I when the machine placement accuracy is at 100 percent, due to the features acting discrete for this phase, while Gaussian naïve Bayes performs best for continuous normally distributed features.

As shown in the results, the CLU model significantly improves upon the baseline in almost every trial. The most noticeable trials where the baseline performs better than CLU are the first trials of Phases II and III. There are two important reasons why this behavior is observed. First, the baseline model for uncertainty tends to be biased towards underconfidence, meaning that its reported probability of correctness tends to be much less than the machine placement accuracy in general. This is also true for higher values of tolerance, where classifier accuracy is expected to be high. Because Phases II and III induce a relatively high amount of concept drift, the machine placement accuracy suffers greatly in the first trial of these phases. As the baseline is biased towards underconfidence, any significant fall in accuracy will cause it to exhibit a higher $1 - MAE$. The second reason that the baseline model outperforms CLU in these particular trials is CLU is unable to detect concept drift from first-class features of the data. As it is driven by feedback of correctness, it must observe that its current policy is inaccurate by observing that the incorrect machine placement was corrected by the human after entering a new phase of drift. However, as the machine placement accuracy improves incrementally in the subsequent trials of Phases II and III, the CLU model is able to very quickly outperform the baseline by accounting for the baseline’s underconfidence by taking into account the incorrect machine placements. Phases IV and V do not generally induce as much concept drift, as

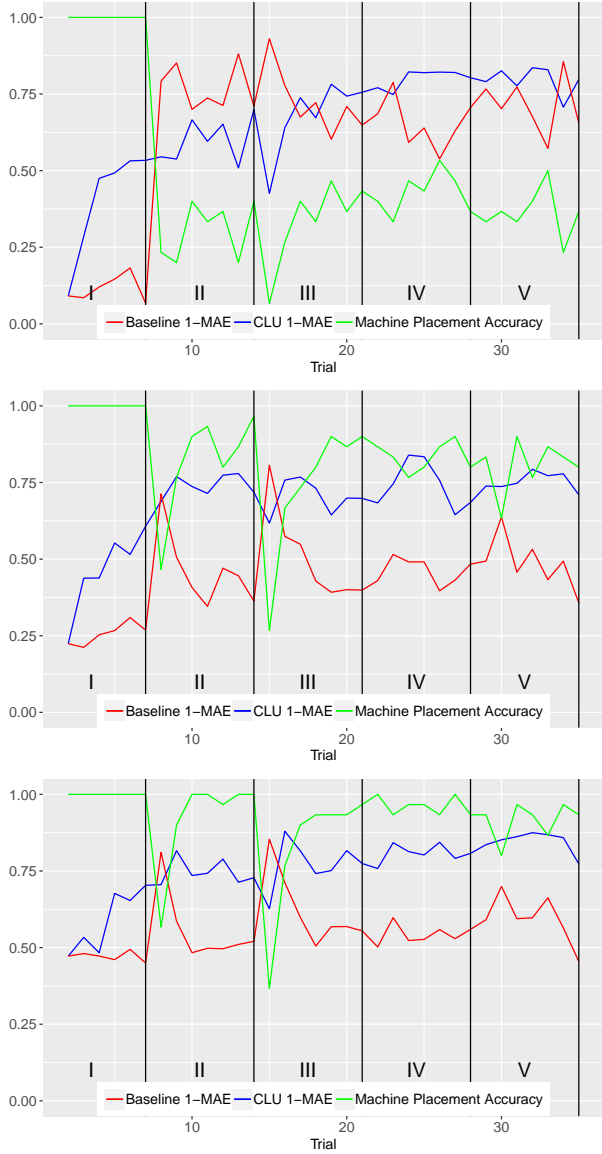


Figure 8. Results comparing the baseline uncertainty model to the CLU model for tolerances 0.02, 0.1, and 0.2 (respectively from top to bottom). Results are plotted for each trial across all realizations. The green plot shows the average accuracy of machine placement.

indicated by the machine placement accuracy of those trials. Additionally, $1 - MAE$ for CLU in Phase I, which is the simplest phase where machine placement is 100% accurate, is lowest among all phases. This is mainly due to the lack of training information from cold start as well as the fact that the bias of the baseline continues to decrease in this phase, undoing the adjustment of the CLU algorithm. Figure 9 shows results for each trial averaged across all tested tolerances.

Mean and variance of precision of confidence for various tol-

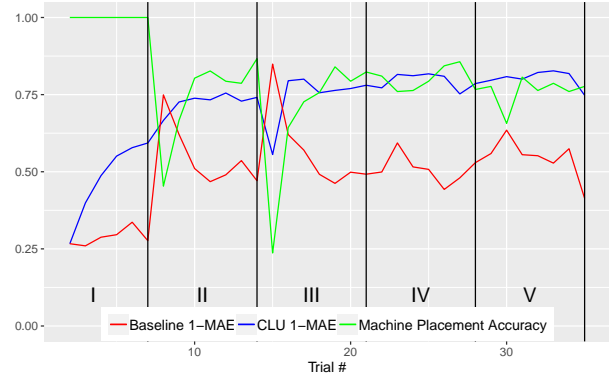


Figure 9. Results as shown in Figure 8, but averaged across all explored tolerances.

Table 1. 1 - MAE for CLU and Baseline models. The mean and variance across all trials and realizations for several tolerances are shown.

Tolerance	1 - MAE			
	CLU		Baseline	
	Mean	Variance	Mean	Variance
0.02	0.663	0.03	0.609	0.0614
0.04	0.701	0.0274	0.507	0.0382
0.06	0.701	0.0276	0.444	0.0241
0.08	0.715	0.0268	0.427	0.0202
0.1	0.686	0.0161	0.44	0.0167
0.12	0.73	0.0162	0.463	0.0154
0.14	0.746	0.0143	0.485	0.0127
0.16	0.733	0.0136	0.509	0.0116
0.18	0.735	0.014	0.537	0.0105
0.2	0.757	0.011	0.56	0.00898

erances are shown in Table 1. These values were calculated across all trials of all realizations of experiments. As shown, CLU is able to improve upon the baseline model by up to 67% and exhibited a 46% average improvement overall. In general, better CLU performance trended towards tolerance values of 0.06-0.16, where the machine placement accuracy was expected to be moderate and the baseline performed most poorly.

9. Conclusions and Future Work

By considering feedback on machine correctness within an iterative IML paradigm, it has been shown that uncertainty modeling can greatly improve upon a baseline that only considers the input data model. A novel closed-loop uncertainty model that exemplifies this improvement was presented. This implementation uses reinforcement learn-

ing to incorporate feedback on machine correctness and adjust the bias of a more traditional uncertainty model. A simple yet effective threshold selection problem, which exhibits many of the important characteristics of IML such as concept drift and subjectivity, was presented and used for experimentation.

The work presented in this paper is meant to demonstrate the concept of modeling and evaluating uncertainty for IML problems. Though the threshold selection problem is interesting and useful for this type of demonstration, the problem itself is relatively simple in terms of classification and feature space. A more complex problem would most likely necessitate a more complex feature space and thus a finer-grained state-action space requiring higher-level optimization. Our group is currently working to extend this research towards the difficult application of geographic region digitization. In these more complex problems, it is mainly the cognitive load of the human analyst that must be taken into consideration. Too little cognitive load, and the machine model is slowing the analyst down; too much, and the analyst becomes overwhelmed and will most likely abandon the machine teammate. As such, we are currently exploring more complex methodologies that analyze uncertainty modeling as a control process with thresholded failure at every iteration.

The detection of concept drift, as in [Du et al. \(2014\)](#), could greatly improve the precision of uncertainty models for iterative streaming problems. We are currently working towards incorporating active learning methods for detecting drift into the closed-loop uncertainty model.

10. Acknowledgements

The authors would like to thank Dr. Jaelle Scheuerman and Dr. Abdul Rahman AlSamman for their recommendations regarding the presentation of this work. This work was sponsored by the U.S. Naval Research Laboratory's Karle Fellowship.

11. Continuation of Research

11.1. Convergence of CLU

The question of the convergence of CLU to well calibrated probabilities is difficult to answer. We know that Q-learning converges to the optimal action-values provided state-action values are repeatedly sampled, and that the state-action space is discrete ([Watkins and Dayan, 1992](#)). What this means is that CLU will converge to an optimal policy that guarantees the highest possible reward, however this is heavily dependent on our definition of the MDP.

If the MDP is defined in such a way that gives higher rewards for a poorer performing uncertainty model, then the result

will be CLU converging to poor uncertainty values, and furthermore if the reward function is essentially set equal to a random value, then the number of trials needed to converge will be prohibitively large. In addition, if the state-action space is poorly defined, too large, or even too small (as demonstrated in [section 6.5](#)), the performance of CLU suffers and the convergence to the optimal policy slows.

In addition, the question of whether CLU converges may be dependent on the choice for the baseline uncertainty model. As we will see in [section 11.2](#), there are several other baseline models that could be conceived that would benefit from the methods imbedded in CLU, however it is also possible to conjure up a baseline model that in no way benefits from CLU. That being said, we have not yet found a baseline model that is adversely affected by CLU.

It is however the view of the researchers that, given a properly defined MDP, CLU has the potential to converge to well calibrated uncertainty values for a wide variety of baseline uncertainty models.

11.2. Alternative Baseline Models

One question that arises from this research is how other types of baseline models perform alongside CLU. In order to investigate this, we tried three different baseline models. This was done not only to check assumptions of the naïve Bayes baseline model by drawing comparisons, but also to investigate the extent to which CLU can be used to improve a traditional uncertainty model.

In the *KDE model* we investigated a Kernel Density Estimation (KDE) technique for the distributions used to calculate the posterior probabilities in naïve Bayes. This method, as discussed in [section 4](#), provides a useful means for which to convert distributions from discrete variables into continuous probability distributions.

The *Induction Model* calculates confidence of each machine placement by equating it to the accuracy of the previous trial. What this means is that if the previous trial had a correct placement (i.e. one in which the machine placement is within a tolerance of the human placement), then the confidence value of the current trial is 1, and if the previous trial had an incorrect placement then the confidence value of the current trial is 0.

A *constant model*, in which all confidence values are set to 0.5, was used as a means to test the basic assumption that the NB model can provide more information about the uncertainty than assuming the odds of a correct classification was a coin flip. In addition, we will see that this model provides evidence that CLU can operate as a stand alone uncertainty model.

It should be noted that the parameters set during the sensitiv-

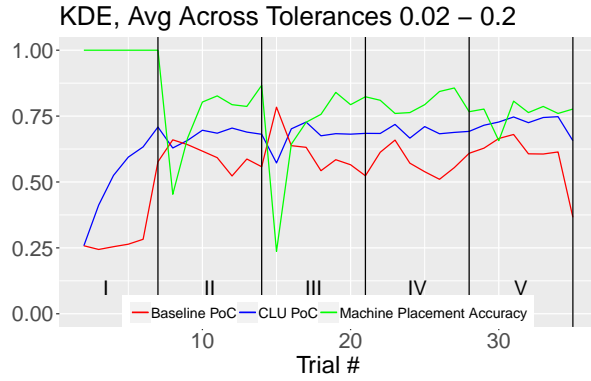


Figure 10. Results as shown in Figure 9, but using KDE as the baseline uncertainty model.

ity analysis discussed in section 6.5 are held fixed throughout this process of testing other baselines. In practice, once baseline model is chosen, a new sensitivity analysis should be conducted.

11.2.1. KERNEL DENSITY ESTIMATION

The advantage to using KDE is it allows for a way to interpret discrete variables in a continuous fashion, which is particularly advantageous in the early phases of each realization where there is much less complexity.

We can see that in figure 10 KDE out performs, in some trials, the Gaussian naïve Bayes model outlined in section 6.1, however the CLU model performed somewhat worse. That being said, CLU does still make a noticeable improvement to the baseline in this case, which further gives evidence to the power of CLU.

11.2.2. INDUCTION

We define an induction baseline uncertainty model as one in which the value for confidence is defined as 1 if the previous trial had a correct classification, and a 0 if the previous trial had an incorrect classification. That is,

$$C_t = \begin{cases} 1 & \text{if } p_{t-1} = 1 \\ 0 & \text{if } p_{t-1} = 0 \end{cases} \quad (20)$$

In practice, there is always a probability of an incorrect classification that is neither 0 nor 1. As a result, this model is inherently either as over confident as possible, or as under confident as possible.

The motivation for this model was to see if a baseline that gave no information about the present trial was still capable of being improved upon using CLU. We can see from figure 11 that this baseline model results in confidence values that

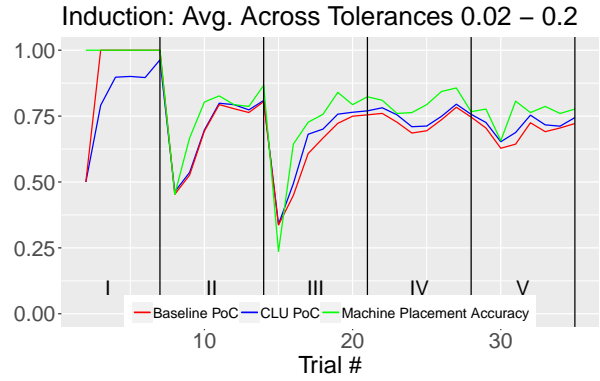


Figure 11. Results as shown in Figure 9, but using induction as the baseline uncertainty model.

are highly susceptible to concept drift, as we can see at the start of phases II and III. This is likely due to the fact that, as per equation 20, confidence is a function of the accuracy of the classifier. In addition, we can see that CLU does not do much in the way of improving the performance. In fact, the induction model slightly out performs CLU in phase I (while CLU warms up), and from then on CLU and this baseline stay pretty much on par with one another. However, it is possible that CLU could begin to out perform the baseline if there were a higher number of trials in each phase.

One possible reason for this is the fact that this baseline does not indicate much information in our definition of the state space. It is not clear whether or not a model similar to CLU could still improve upon this induction baseline if there is a reformulation of the state space to include information that is not encoded in successive baseline confidence values.

11.2.3. CONSTANT

A baseline model that assumes every classification has a confidence of 0.5 is essentially equivalent to assuming the classification is as good as a coin flip. This model was used as a baseline, because we began to wonder about the implications of some of the baseline confidence values below 0.5 found using the gaussian naïve bayes model. In figure 8 and figure 9, during phase I the confidence values are quite low, however this is also the phase where the classifier performed at essentially perfect accuracy. Intuitively, this makes little sense, which is what motivated this constant baseline model.

We can see from figure 12 that in this case CLU does in fact out perform the baseline. This suggests that CLU could operate as a relatively decent uncertainty model, without having the advantage provided by a nuanced baseline model.

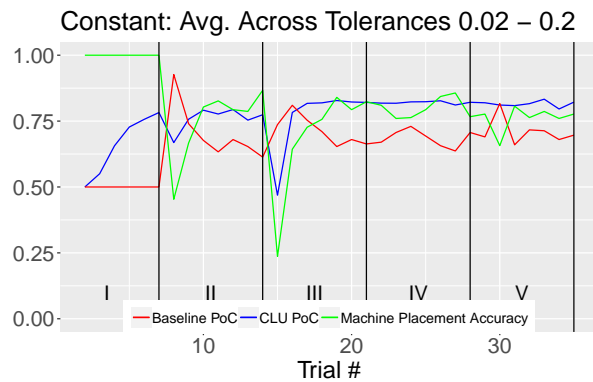


Figure 12. Results as shown in Figure 9, but using the constant model as the baseline uncertainty model.

11.3. Subjectivity in Trials

When the initial 30 realizations of the experiment were conducted, the analyst made the subjective choice on the first 22 trials that a signal without noise is no longer considered high when it falls below 0.5. This subjective choice was set to 0.75 for the last 8 realizations, for the simple reason of giving a diversity of experimentation.

One result of this was that there was a small set of realizations under the latter subjective choice, i.e. where a signal under $y=0.75$ is no longer considered high. During those last 8 realizations, at trial number 30 there was a slightly higher than average value for the magnitude of the noise m . This means that in trial 30 there was a decrease in classification accuracy in those final 8 realizations. Because in the first 22 realizations the classification accuracy was quite high in phase 5, trial 30 showed a significant drop in classification accuracy as a result of those final 8 realizations.

References

- Abdar, M., Pourpanah, F., Hussain, S., Rezazadegan, D., Liu, L., Ghavamzadeh, M., Fieguth, P., Cao, X., Khosravi, A., Acharya, U. R., et al. (2021). A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion*, 76:243–297.
- Aggarwal, C. C., Kong, X., Gu, Q., Han, J., and Philip, S. Y. (2014). Active learning: A survey. In *Data Classification*, pages 599–634. Chapman and Hall/CRC.
- Bellemare, M. G., Dabney, W., and Munos, R. (2017). A distributional perspective on reinforcement learning. In *International Conference on Machine Learning*, pages 449–458. PMLR.
- Bhatt, U., Antorán, J., Zhang, Y., Liao, Q. V., Sattigeri, P., Fogliato, R., Melançon, G., Krishnan, R., Stanley, J., Tickoo, O., et al. (2021). Uncertainty as a form of transparency: Measuring, communicating, and using uncertainty. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, pages 401–413.
- Clements, W. R., Van Delft, B., Robaglia, B.-M., Slaoui, R. B., and Toth, S. (2020). Estimating risk and uncertainty in deep reinforcement learning. In *Uncertainty and Robustness in Deep Learning Workshop at International Conference on Machine Learning*.
- Du, L., Song, Q., and Jia, X. (2014). Detecting concept drift: an information entropy based method using an adaptive sliding window. *Intelligent Data Analysis*, 18(3):337–364.
- Fails, J. A. and Olsen Jr, D. R. (2003). Interactive machine learning. In *Proceedings of the 8th international conference on Intelligent user interfaces*, pages 39–45.
- Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. (2017). On calibration of modern neural networks. In *International conference on machine learning*, pages 1321–1330. PMLR.
- Hasselt, H. V. (2012). Reinforcement learning in continuous state and action spaces. In *Reinforcement learning*, pages 207–251. Springer.
- Hora, S. C. (1996). Aleatory and epistemic uncertainty in probability elicitation with an example from hazardous waste management. *Reliability Engineering & System Safety*, 54(2-3):217–223.
- Hüllermeier, E. and Waegeman, W. (2021). Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine Learning*, 110(3):457–506.
- Jiang, H., Kim, B., Guan, M., and Gupta, M. (2018). To trust or not to trust a classifier. *Advances in neural information processing systems*, 31.
- Jiang, L., Liu, S., and Chen, C. (2019). Recent research advances on interactive machine learning. *Journal of Visualization*, 22(2):401–417.
- John, G. H. and Langley, P. (2013). Estimating continuous distributions in bayesian classifiers. *arXiv preprint arXiv:1302.4964*.
- Kabra, M., Robie, A. A., Rivera-Alba, M., Branson, S., and Branson, K. (2013). Jaaba: interactive machine learning for automatic annotation of animal behavior. *Nature methods*, 10(1):64–67.
- Kaplan, L., Cerutti, F., Sensoy, M., Preece, A. D., and Sullivan, P. (2018). Uncertainty aware ai ml: Why and how.

-
- Klås, M. and Vollmer, A. M. (2018). Uncertainty in machine learning applications: A practice-driven classification of uncertainty. In *International conference on computer safety, reliability, and security*, pages 431–438. Springer.
- Krawczyk, B. and Wozniak, M. (2015). Weighted naive bayes classifier with forgetting for drifting data streams. In *2015 IEEE International conference on systems, man, and cybernetics*, pages 2147–2152. IEEE.
- Li, M. and Sethi, I. K. (2006). Confidence-based active learning. *IEEE transactions on pattern analysis and machine intelligence*, 28(8):1251–1261.
- Lu, J., Liu, A., Song, Y., and Zhang, G. (2020). Data-driven decision support under concept drift in streamed big data. *Complex & Intelligent Systems*, 6(1):157–163.
- Michael, C. J., Acklin, D., and Scheuerman, J. (2020). On interactive machine learning and the potential of cognitive feedback. *arXiv preprint arXiv:2003.10365*.
- Michael, C. J., Dennis, S. M., Maryan, C., Irving, S., and Palmsten, M. L. (2019). A general framework for human-machine digitization of geographic regions from remotely sensed imagery. pages 259–268.
- Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.
- Monarch, R. M. (2021). *Human-in-the-Loop Machine Learning: Active learning and annotation for human-centered AI*. Simon and Schuster.
- Mosqueira-Rey, E., Hernández-Pereira, E., Alonso-Ríos, D., Bobes-Bascarán, J., and Fernández-Leal, Á. (2022). Human-in-the-loop machine learning: a state of the art. *Artificial Intelligence Review*, pages 1–50.
- Nikolov, N., Kirschner, J., Berkenkamp, F., and Krause, A. (2018). Information-directed exploration for deep reinforcement learning. *International Conference on Learning Representations (ICLR)*.
- Pronk, V., Gutta, S. V., and Verhaegh, W. F. (2005). Incorporating confidence in a naive bayesian classifier. In *International Conference on User Modeling*, pages 317–326. Springer.
- Provost, F. J., Fawcett, T., Kohavi, R., et al. (1998). The case against accuracy estimation for comparing induction algorithms. In *ICML*, volume 98, pages 445–453.
- Sahinidis, N. V. (2004). Optimization under uncertainty: state-of-the-art and opportunities. *Computers & Chemical Engineering*, 28(6-7):971–983.
- Schlimmer, J. C. and Granger, R. H. (1986). Incremental learning from noisy data. *Machine learning*, 1:317–354.
- Sutton, R. S. and Barto, A. G. (1998). Reinforcement learning: an introduction mit press. *Cambridge, MA*, 22447.
- Teso, S. and Vergari, A. (2022). Efficient and reliable probabilistic interactive learning with structured outputs.
- Wang, G., Li, W., Aertsen, M., Deprest, J., Ourselin, S., and Vercauteren, T. (2019). Aleatoric uncertainty estimation with test-time augmentation for medical image segmentation with convolutional neural networks. *Neuro-computing*, 338:34–45.
- Watkins, C. J. and Dayan, P. (1992). Q-learning. *Machine learning*, 8(3):279–292.
- Zhang, H. (2004). The optimality of naive bayes. *Aa*, 1(2):3.