SANDIA REPORT
SAND2022-xxxx
Printed

Sandia National Laboratories

A Decision Theoretic Approach To Optimizing Machine Learning Decisions with Prediction Uncertainty

Richard V. Field Jr. and Michael C. Darling

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185
Livermore, California 94550

Issued by Sandia National Laboratories, operated for the United States Department of Energy by National Technology & Engineering Solutions of Sandia, LLC.

Available to DOE and DOE contractors from

U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@osti.gov
Online ordering: http://www.osti.gov/scitech

Available to the public from

U.S. Department of Commerce
National Technical Information Service
5301 Shawnee Road
Alexandria, VA 22312

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.gov
Online order: https://classic.ntis.gov/help/order-methods

ABSTRACT

While the use of machine learning (ML) classifiers is widespread, their output is often not part of any follow-on decision-making process. To illustrate, consider the scenario where we have developed and trained an ML classifier to find malicious URL links. In this scenario, network administrators must decide whether to allow a computer user to visit a particular website, or to instead block access because the site is deemed malicious. It would be very beneficial if decisions such as these could be made automatically using a trained ML classifier. Unfortunately, due to a variety of reasons discussed herein, the output from these classifiers can be uncertain, rendering downstream decisions difficult. Herein, we provide a framework for: (1) quantifying and propagating uncertainty in ML classifiers; (2) formally linking ML outputs with the decision-making process; and (3) making optimal decisions for classification under uncertainty with single or multiple objectives.

ACKNOWLEDGMENT

The authors thank J.D. Doak, Mark Smith, David Stracuzzi, and countless others for providing helpful discussion and comments throughout the life of this project.

This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government. This work was supported by the Laboratory Directed Research and Development program at Sandia National Laboratories, a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

CONTENTS

LIST OF FIGURES

1. INTRODUCTION

The decision science (DS) method of utility maximization determines the course of action that maximizes expected utilities while accounting for the probability that an event will occur. As DS methods were developed under the assumption that probability values are deterministic and well-calibrated, they do not consider uncertainty in these estimates. Therefore, to use such approaches for high-consequence decisions with probabilities estimated from ML models, the uncertainty in these estimates must be quantified.

Example 1.0.1. Consider an ML model tasked with identifying malicious websites. Allowing a connection to a malicious website could result in a ransomware attack; blocking a legitimate URL merely results in lost productivity for an inconvenienced user. The costs of a false negative, predicting that a malicious website is legitimate, are therefore potentially magnitudes larger than those of a false positive. This domain presents a further complexity: as with many problems solved with machine learning, the true probabilistic structure is unknowable. We can, however, understand the reliability of ML model estimates by quantifying uncertainty.

In this report we formally link ML classification (with two or more classes) with decision theory; this required modifications to standard decision science techniques to handle uncertainty in probabilistic class labels. We develop a framework for making optimal decisions for classification under uncertainty with single or multiple objectives; our approach keeps decision-making independent from ML training, meaning that decision-makers do not have to be ML experts. We also show how to make decisions that are optimal per instance, instead of optimal on average across a population as is the case for cost-sensitive machine learning; this is especially important for high-consequence applications.

This work was developed under the lab-directed research project entitled "Optimizing Machine Learning Decisions with Prediction Uncertainty" [1]. The overall goal of the project was to develop decision-making methods that minimize prediction error costs for any ML model. For each prediction made by the model, the decision-making algorithm would consider an ML model's score, or probability estimate, the uncertainty in that estimate, the consequences associated with available actions, and the costs of further analyses or information gathering.

Traditionally, the ML community, under the heading of cost-sensitive learning, has attempted to minimize expected error costs by incorporating the relative costs of different

error types and class imbalance information into training methods, thus averaging cost over a population of examples. We proposed to focus on the uncertainty estimated for individual predictions, thereby incorporating example-specific information and explicitly weighing whether the ML model is qualified to assess a given example.

The need for this type of work became apparent during the *Multimodal Data Integration Under Uncertainty* LDRD project [2, 3]. Which developed uncertainty analysis methods while applying them to a number of problems including malicious URL detection [4], multi-source image analysis [5], and seismic onset detection [6]. It became increasingly evident that uncertainty quantification (UQ) for ML predictions provided important information to consider for decision-making.

We note that ML classification and decision-making are not synonymous. While ML maps observed data to unobservable properties of interest, decision science defines a decision as an irrevocable allocation of resources. Therefore, the aim of this work is to decide the optimal course of action given information provided by and about an ML model's output. In this project, we assume that the ML model is already given. In other words, we are already past the point of changing the model architecture or retraining it. This has the advantage of clearly separating the problem of model design from the problem of using ML models as an aid in decision making.

2. DECISION-MAKING WITH ML CLASSIFIER PREDICTIONS

We first provide some basics of ML classification in Section 2.1. A decision-theoretic approach based on minimizing expected loss is then presented in Section 2.1.1 for optimal actions based on ML model outputs. The effects of uncertainty are not considered until Section 3.

2.1. Basics of machine learning classification

In this section, we review some basics of machine learning (ML) classification and introduce notion; we follow the notation used by Flach [7].

The objects of interest in ML are referred to as *instances*; the set of all possible instances $\mathcal{X}$ is called the *instance space*. Strictly speaking, ML classifiers almost always work on attributes or *features* of $\boldsymbol{x}$ and not on the instance itself. To simplify notation we do not worry about this distinction and just assume that each $\boldsymbol{x} \in \mathbb{R}^d$ is a real-valued vector of $d > 0$ features. In URL classification introduced in Example 1.0.1, $\boldsymbol{x}$ would contain features of a URL, such as the domain name or the number of characters in the URL.

Next let $\mathcal{C}$ denote the label or *class space*; each member $C \in \mathcal{C}$ is a *class*, and we use symbol $\kappa = |\mathcal{C}|$ to denote the number of classes. In URL classification, $\mathcal{C} = \{\texttt{benign}, \texttt{malicious}\}$ and $\kappa = 2$; this is binary classification. A *classifier* is a mapping $\hat{c}\colon \mathcal{X} \to \mathcal{C}$; for arbitrary instance $\boldsymbol{x}$, $\hat{c}(\boldsymbol{x})$ is an estimate of the true but unknown label function $c(\boldsymbol{x})$. Herein, we consider scoring classifiers that output a probability vector over the classes, i.e., mappings $\hat{\mathbf{p}}\colon \mathcal{X} \to [0,1]^\kappa$. The boldface notation indicates that the scoring classifier outputs a vector $\hat{\mathbf{p}}(\boldsymbol{x}) = (\hat{p}_i(\boldsymbol{x}), \ldots, \hat{p}_\kappa(\boldsymbol{x}))^T$, where $\hat{p}_i(\boldsymbol{x}) \geq 0$ is the probability assigned to class C_i for instance $\boldsymbol{x}$, and $\hat{p}_1(\boldsymbol{x}) + \cdots + \hat{p}_\kappa(\boldsymbol{x}) = 1$.

We note that $\hat{\mathbf{p}}(\boldsymbol{x})$ is an estimate of the true but unknown probability distribution $\mathbf{p}(\boldsymbol{x})$. For the special case of binary classification, i.e., $\kappa = 2$, $\hat{p}_2(\boldsymbol{x}) = 1 - \hat{p}_1(\boldsymbol{x})$ and only one symbol is needed; we then use $\hat{p}(\boldsymbol{x})$ and $1 - \hat{p}(\boldsymbol{x})$ to represent the probability that instance $\boldsymbol{x}$ is of class C_1 and C_2, respectively.

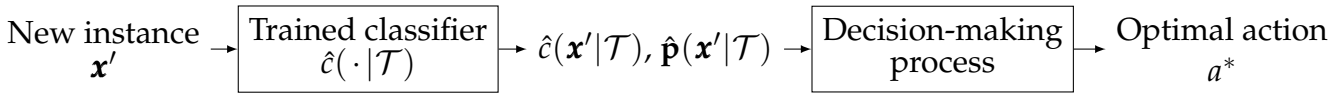

Figure 2-1. A decision about x' based on ML model prediction.

The predicted class label $\hat{c}(\boldsymbol{x})$ is completely defined by $\hat{\mathbf{p}}(\boldsymbol{x})$. For example, for binary classification, we have

$$\hat{c}(\boldsymbol{x}) = \begin{cases} C_1 & \hat{p}(\boldsymbol{x}) \leq t \\ C_2 & \hat{p}(\boldsymbol{x}) > t \end{cases} \tag{2.1}$$

where $t \in (0,1)$ is a threshold parameter (usually equal to $1/2$). For $\kappa > 2$ classes, this generalizes to

$$\hat{c}(\boldsymbol{x}) = C_{i^*}, \text{ where } i^* = \underset{1 \leq i \leq \kappa}{\arg\max}\, \hat{p}_i(\boldsymbol{x}). \tag{2.2}$$

Lastly, let $\mathcal{T} = \{(\boldsymbol{x}, c(\boldsymbol{x}))\}$ denote the *training set*, that is, a collection of labeled instances used to calibrate or train the ML classifier. The outputs from a trained ML model depend on the training data used; we will therefore denote the predicted class label and probability scores by $\hat{c}(\boldsymbol{x}|\mathcal{T})$ and $\hat{\mathbf{p}}(\boldsymbol{x}|\mathcal{T})$, respectively.

2.1.1. Decision theory applied to ML classification

Suppose now that, given a previously unseen instance $\boldsymbol{x}'$, there exists some action or decision that needs to be made about $\boldsymbol{x}'$, and the purpose of the trained classifier is to aid in the decision making process. The objective is then to make the best decision possible about $\boldsymbol{x}'$ based on supporting information; decision theory provides a formal framework for doing this. Refer to Fig. 2-1; the remainder of this section provides specifics about the decision-making process.

A decision problem exists when there is a choice of possible actions to take; the consequence of these actions depends on the (unknown) truth, typically referred to as the "state of nature" in the decision theory literature [8]. The general problem in decision theory has four ingredients [9].

1. A collection of possible actions to take.
2. A collection of all possible states of nature.
3. A loss function.

		Nature, $\mathcal{C}$	
		`benign`	`malicious`
Action, $\mathcal{A}$	`allow`	0	20
	`block`	5	1

Figure 2-2. Example loss function for URL classification.

4. The optimal action.

We define

$$\mathcal{A} = \{\text{all candidate actions}\} = \{a_i\}$$

to be the collection of candidate actions that a decision-maker can take. For the URL example problem mentioned introduced in Example 1.0.1, $\mathcal{A} = \{\texttt{allow},\texttt{block}\}$.

In the context of ML classification of instance $\boldsymbol{x}'$, the (unknown) state of nature is simply the true label $c(\boldsymbol{x}')$. Hence, the collection of all possible states of nature is identical to the label space, $\mathcal{C}$, defined above; the probability of these states being true is $\mathbf{p}(\boldsymbol{x}')$.

We define a loss function, $L\colon \mathcal{A} \times \mathcal{C} \to [0,\infty)$, to quantify the consequences of each action, under each state of nature. Hence $L(a_i, C_j) \geq 0$ is the consequence for taking action a_i, assuming the true label is $c(\boldsymbol{x}') = C_j$. One possible form for the loss function is given by

$$L(a_i, C_j) = \gamma(a_i) + \psi(a_i, C_j), \tag{2.3}$$

where $\gamma(a_i) \geq 0$ quantifies the "cost" of taking action a_i, and $\psi(a_i, C_j) \geq 0$ represents the penalty associated with taking action a_i when the true label of $\boldsymbol{x}'$ is C_j.

Example 2.1.1. The loss function defined by Eq. (2.3) is illustrated by Fig. 2-2 for the URL classification example introduced in Example 1.0.1; numerical values are made-up for clarity. If a decision-maker allows a benign URL, there is zero cost and zero penalty. Blocking a malicious URL, however, has a cost of one, but no penalty. The off-diagonal terms in the table correspond to scenarios where there are penalties, and the penalty of allowing a malicious URL is larger than blocking a benign one. ◇

By definition, the loss is a random variable because the state of nature, i.e., the true class label of $\boldsymbol{x}'$, is uncertain. We can define the expected loss of action a_i as

$$\ell(a_i) = \mathbb{E}[L(a_i, \mathcal{C})] = \sum_{j=1}^{\kappa} L(a_i, C_j) p_j(\boldsymbol{x}') \approx \sum_{j=1}^{\kappa} L(a_i, C_j) \hat{p}_j(\boldsymbol{x}' | \mathcal{T}), \tag{2.4}$$

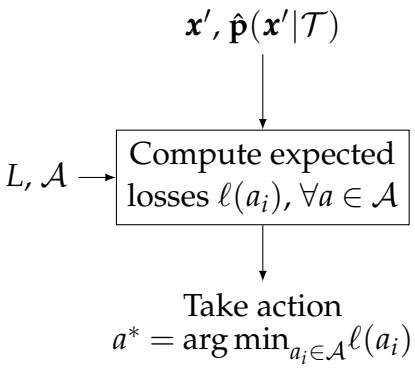

Figure 2-3. The decision-making process for ML classification.

which depends on the true and unknown probability vector $\mathbf{p}(\boldsymbol{x}')$. In practice, we approximate $\mathbf{p}(\boldsymbol{x}')$ with $\hat{\mathbf{p}}(\boldsymbol{x}'|\mathcal{T})$, the output from the ML classifier.

Action a_i is optimal if and only if $\ell(a_i) \leq \ell(a_j)$ for all $i \neq j$; in this case, we denote the optimal action by a^*, that is

$$a^* = \arg\min_{a_i \in \mathcal{A}} \ell(a_i). \tag{2.5}$$

Therefore, the best decision we can make about $\boldsymbol{x}'$ given the available information we have is to take action a^*. The decision process is illustrated as a flowchart by Fig. 2-3.

Example 2.1.2. For example, with the loss function discussed in Example 2.1.1, we have

$$\begin{aligned}\ell(\texttt{allow}) &= 20\left(1 - \hat{p}(\boldsymbol{x}'|\mathcal{T})\right)\\ \ell(\texttt{block}) &= 5\,\hat{p}(\boldsymbol{x}'|\mathcal{T}) + \left(1 - \hat{p}(\boldsymbol{x}'|\mathcal{T})\right)\end{aligned}$$

where $\hat{p}(\boldsymbol{x}'|\mathcal{T}) = \mathbb{P}(\texttt{benign})$, so that $a^* = \texttt{allow}$ when $\hat{p}(\boldsymbol{x}'|\mathcal{T}) \geq 19/24$. ◇

We note that taking the action that minimizes the expected loss defined by Eq. (2.4) is a common approach in decision theory, but it is not the only approach. It is possible to base decisions on alternatives to the expected loss.

3. DECISION-MAKING WITH ML MODELS UNDER UNCERTAINTY

In this chapter, we propose a unified framework for decision-making based on the outputs from ML classifiers that is robust to a variety of sources of uncertainty in ML models. In Section 3.1 we present an overview of uncertainty in ML classification.

3.1. Uncertainty quantification for machine learning

There are a variety of sources of uncertainty that impact the outputs of ML models [2, 10]. We describe some of the sources below; this is an incomplete list.

3.1.1. Uncertainty from the data

There can be uncertainty in the output of a ML classifier resulting from one or more of the following sources related to the data.

1. *Pre-processing and/or measurement error*

   Errors in pre-processing of the data, such as improper filtering, can lead to uncertainty in ML predictions. In addition, one of more instance $\boldsymbol{x}$ and/or corresponding label $c(\boldsymbol{x})$ from the training data may be missing or corrupted with noise; this can occur by accident or be the result of nefarious activity by an adversary.

2. *Limited training data*

   When the number of training data is small or include redundancies, the training step can be inaccurate; this can lead to uncertainty in the predicted class labels. One way to account for this is based on bootstrap sampling (refer to Fig. 3-1):

   a) Draw samples with replacement from the training set; call this set $\mathcal{T}_i \subseteq \mathcal{T}$.

   b) Train the ML classifier on $\mathcal{T}_i$.

   c) Make prediction $\hat{\mathbf{p}}(\boldsymbol{x}'|\mathcal{T}_i)$ about the unknown instance $\boldsymbol{x}'$.

   d) Repeat the previous 3 steps $b > 1$ times.

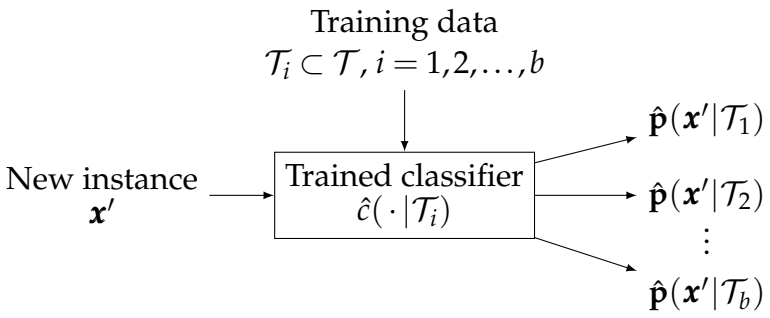

Figure 3-1. ML model prediction with uncertainty.

The output from this procedure will produce b different predictions, which can be interpreted as b samples of $\hat{\mathbf{p}}(\boldsymbol{x}'|\mathcal{T})$. These samples represent uncertainty due to the training process.

3. *Extrapolation error*

   If a new previously unseen instance $\boldsymbol{x}'$ is "far" from the centroid of the training instances, the trained model applied to this instance provides an extrapolation. In such cases, the predicted class label $\hat{c}(\boldsymbol{x}'|\mathcal{T})$ will exhibit uncertainty because the accuracy of the model in this region of the feature space is unknown. One approach to quantifying the distance that $\boldsymbol{x}'$ is from the training instances is the Local Outlier Factor (LOF) [11], which we will discuss in Section 3.3.2.

3.1.2. Uncertainty from the model

There can be uncertainty in the output of a ML classifier resulting from one or more of the following sources related to the ML model.

1. *Model form uncertainty*

   There are many algorithms for binary classification, such as decision trees and linear classifiers, and each algorithm requires the user to set various parameters; for example, the maximum depth of a tree. In general, each of these types of algorithms can be trained and used to make predictions, and different models trained with identical data may yield different predictions. For example, let $\hat{c}_1$ and $\hat{c}_2$ two different classifiers both trained with $\mathcal{T}$. Given a new instance $\boldsymbol{x}'$, its predicted class label is uncertain if $\hat{c}_1(\boldsymbol{x}'|\mathcal{T}) \neq \hat{c}_2(\boldsymbol{x}'|\mathcal{T})$. If the predicted class $\hat{c}(\boldsymbol{x}'|\mathcal{T})$ is the same regardless of the algorithm used and over a wide variety of parameter values, then this source of uncertainty can be ignored. In most cases, this is not true.

2. *Feature selection*

   Feature selection is a extremely important part of ML classification; no classifier will perform well if features are selected poorly. Ideally, the features are defined in such a way that, as the number of training data increases without bound, the true label function can be learned: $\hat{c}(\boldsymbol{x}'|\mathcal{T}) \to c(\boldsymbol{x}')$. Unfortunately, in general there is no formal way to select or design features that can guarantee this result. Further, predictions on a new instance based on different features will, in general, produce different results. Feature sets can be non-unique: ML model performance is identical under two or more feature sets. This is therefore another source of uncertainty in ML outputs.

3.2. Propagation to output class

Recall that, for new instance $\boldsymbol{x}'$, we have the following:

1. The true (unknown) probability vector $\mathbf{p}(\boldsymbol{x}') = (p_1(\boldsymbol{x}'),\ldots,p_\kappa(\boldsymbol{x}'))$, where $p_i(\boldsymbol{x}') = \mathbb{P}(c(\boldsymbol{x}') = C_i)$ is the probability that the true class of $\boldsymbol{x}'$ is C_i; and
2. Vector $\hat{\mathbf{p}}(\boldsymbol{x}'|\mathcal{T}) = (\hat{p}_1(\boldsymbol{x}'|\mathcal{T}),\ldots,\hat{p}_\kappa(\boldsymbol{x}'|\mathcal{T}))$, the output from the trained ML classifier, where each $\hat{p}_i(\boldsymbol{x}'|\mathcal{T})$ is an approximation to $p_i(\boldsymbol{x}')$.

Due to the variety of sources as discussed in Section 3.1, the output probability scores from an ML classifier cannot be treated as a deterministic vector and, instead, should be treated as a random vector, that is, $\hat{\mathbf{p}}(\boldsymbol{x}'|\mathcal{T})$ is a vector of $\kappa = |\mathcal{C}|$ continuous random variables, each with support $[0,1]$. To further emphasize that $\hat{\mathbf{p}}(\boldsymbol{x}'|\mathcal{T})$ is a random vector, we introduce a new symbol $\mathbf{Q}$, where $Q_i = \hat{p}_i(\boldsymbol{x}'|\mathcal{T}) = \mathbb{P}(\hat{c}(\boldsymbol{x}') = C_i|\mathcal{T})$, $i = 1,\ldots,\kappa$.

Because they represent probabilities, the coordinates of random vector $\mathbf{Q}$ must satisfy $0 \le Q_i \le 1$, $i = 1,\ldots,\kappa$, and $Q_1 + \cdots + Q_\kappa = 1$ almost surely. Hence the support of $\mathbf{Q}$ is equal to the $(\kappa-1)$-simplex in $[0,1]^\kappa$. For example, the support of $\mathbf{Q}$ is a line segment, a triangle, and a tetrahedron for $\kappa = 2$, $\kappa = 3$, and $\kappa = 4$, respectively.

The simple approximation $\mathbf{p}(\boldsymbol{x}') \approx \hat{\mathbf{p}}(\boldsymbol{x}'|\mathcal{T})$ we applied in Sections 2.1 and 2.1.1 is no longer meaningful because the latter term, which comes from a ML classifier, is not deterministic; we must therefore derive a suitable approximation for $\mathbf{p}(\boldsymbol{x}')$. We first consider binary classification ($\kappa = 2$) for simplicity, and then generalize for the case of $\kappa > 2$.

3.2.1. Binary classification

Let $p(\boldsymbol{x}') = \mathbb{P}(c(\boldsymbol{x}') = C_1)$ and $\mathbb{P}(c(\boldsymbol{x}') = C_2) = 1 - p(\boldsymbol{x}')$ denote the probability distribution of the true label. Let random variable $Q = \hat{p}(\boldsymbol{x}'|\mathcal{T}) = \mathbb{P}(\hat{c}(\boldsymbol{x}'|\mathcal{T}) = C_1)$ be the output

from the ML classifier; it follows that $\mathbb{P}(\hat{c}(\boldsymbol{x}'|\mathcal{T}) = C_2) = 1 - Q$. Further let $F(q) = \mathbb{P}(Q \leq q)$ denote the cdf of Q.

Recall the classification rule defined by Eq. (2.2) introduced in Section 2.1. This implies that an appropriate approximation for the probability distribution of the true label that includes uncertainty in the output of the ML model is given by

$$\begin{aligned} \mathbb{P}(c(\boldsymbol{x}') = C_1) &\approx \mathbb{P}(Q \leq t) = F(t) \\ \mathbb{P}(c(\boldsymbol{x}') = C_2) &\approx \mathbb{P}(Q > t) = 1 - F(t). \end{aligned} \tag{3.1}$$

Example 3.2.1. For example, if the output from the ML classifier $\hat{p}(\boldsymbol{x}'|\mathcal{T})$ follows a beta distribution with shape parameters $a, b > 0$, then we approximate the distribution of the true label as

$$\mathbb{P}(c(\boldsymbol{x}') = C_1) \approx F(t) = \frac{\beta(t;a,b)}{\beta(1;a,b)},$$

where

$$\beta(t;a,b) = \int_0^t x^{a-1}(1-x)^{b-1}\,\mathrm{d}x$$

is the incomplete beta function. This case is illustrated by Fig. 3-2. The pdf of the beta distribution is shown in Fig. 3-2(a) assuming $a = b = 3$; the corresponding approximation for $p(\boldsymbol{x}')$ is shown in Fig. 3-2(b). For threshold $t = 2/3$, $\mathbb{P}(c(\boldsymbol{x}') = C_1) \approx 0.8$ and $\mathbb{P}(c(\boldsymbol{x}') = C_2) \approx 0.2$. ◇

Example 3.2.2. If instead the distribution of $\hat{p}(\boldsymbol{x}'|\mathcal{T})$ follows a mixture of n Gaussian random variables with means μ_i, standard deviations $\sigma_i > 0$, and weights $0 \leq w_i \leq 1$, $i = 1, \ldots, n$, such that $w_1 + \cdots w_n = 1$, then we approximate the distribution of the true label as

$$\mathbb{P}(c(\boldsymbol{x}') = C_1) \approx F(t) = \sum_{i=1}^{n} w_i \Phi\left(\frac{t - \mu_i}{\sigma_i}\right),$$

where Φ denotes the cdf of the standard Gaussian random variable. This case is illustrated by Fig. 3-3 assuming $w_2 = \mu_2 = 0.7$, $w_1 = \mu_1 = 0.3$, and $\sigma_1 = \sigma_2 = 0.05$. For threshold $t = 1/2$, $\mathbb{P}(c(\boldsymbol{x}') = C_1) \approx 0.3$ and $\mathbb{P}(c(\boldsymbol{x}') = C_2) \approx 0.7$. ◇

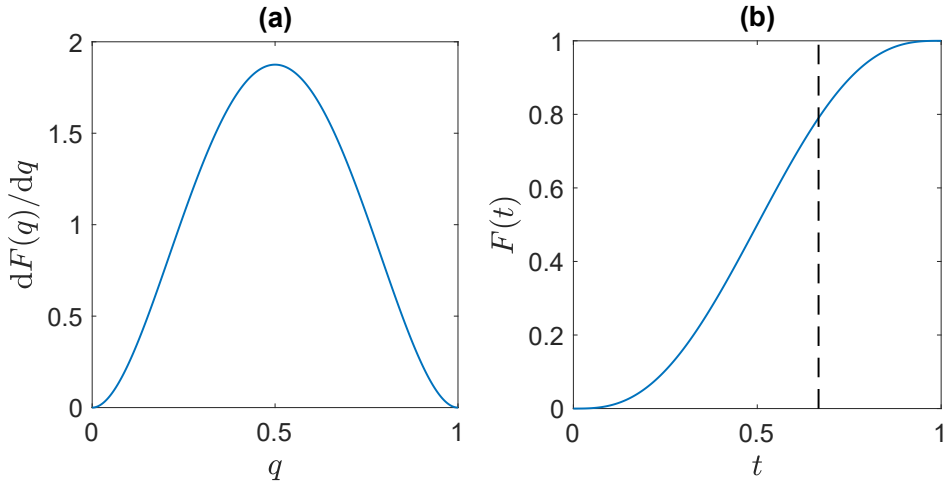

Figure 3-2. Approximation for $p(\boldsymbol{x}') = \mathbb{P}(c(\boldsymbol{x}') = C_1)$ described in Example 3.2.1 assuming the output from the ML model $\hat{p}(\boldsymbol{x}'|\mathcal{T})$ follows a beta distribution. Panel (a) illustrates the pdf of the beta random variable with $a = b = 3$; panel (b) illustrates the approximation $\mathbb{P}(c(\boldsymbol{x}') = C_1) \approx F(t)$.

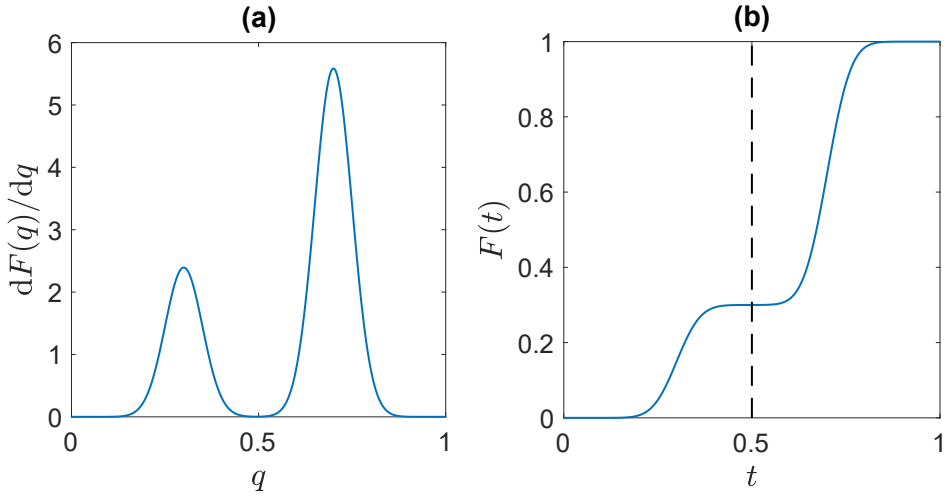

Figure 3-3. Approximation for $p(\boldsymbol{x}') = \mathbb{P}(c(\boldsymbol{x}') = C_1)$ described in Example 3.2.2 assuming the output from the ML model $\hat{p}(\boldsymbol{x}'|\mathcal{T})$ follows a mixture of two Gaussian distributions. Panel (a) illustrates the pdf of the Gaussian mixture; panel (b) illustrates the approximation for $\mathbb{P}(c(\boldsymbol{x}') = C_1) \approx F(t)$.

3.2.2. Multi-class classification

In the general case of $\kappa \geq 2$ classes, we apply Eq. (2.2), the multi-class classification rule, to show that

$$\mathbb{P}(c(\boldsymbol{x}') = C_i) \approx \mathbb{P}(Q_i > Q_j, \forall i \neq j) = \int_{\substack{q_i > q_j \\ \forall i \neq j}} \mathrm{d}F(\mathbf{q}), \tag{3.2}$$

where $F(\mathbf{q}) = \mathbb{P}(Q_1 \leq q_1, \ldots, Q_\kappa \leq q_\kappa)$ is the joint cdf of $\mathbf{Q} = (Q_1, \ldots, Q_\kappa)$, $Q_i = \hat{p}_i(\boldsymbol{x}'|\mathcal{T}) = \mathbb{P}(\hat{c}(\boldsymbol{x}') = C_i|\mathcal{T})$, the (random) output from the ML classifier.

Example 3.2.3. To illustrate this result, we consider a $\kappa = 3$ class example. In this case, the output from the ML classifier $\hat{\mathbf{p}}(\boldsymbol{x}'|\mathcal{T})$ is a random vector with three coordinates, $\mathbf{Q} = (Q_1, Q_2, Q_3)$, where $Q_i = \hat{p}_i(\boldsymbol{x}'|\mathcal{T})$ and $Q_1 + Q_2 + Q_3 = 1$. For calculations, we assume the joint cdf of $\mathbf{Q}$, denoted by $F(\mathbf{q})$, is defined such that

$$\begin{aligned} Q_1 &\sim \text{Beta}(a, b) \\ Q_2|(Q_1 = q_1) &\sim (1 - q_1)\,\text{Beta}(a, b) \\ Q_3|(Q_1 = q_1, Q_2 = q_2) &= 1 - q_1 - q_2. \end{aligned}$$

With this model, Q_1 is a beta random variable with shape parameters $a, b > 0$ and takes values in $(0, 1)$. The conditional random variable $Q_2|(Q_1 = q_1)$ is a beta random variable with identical shape parameters and takes values in $(0, q_1)$; by this construction, we are guaranteed that every sample of (Q_1, Q_2) satisfies $Q_1 + Q_2 \leq 1$. With Q_1 and Q_2 defined, random variable Q_3 is defined so that $Q_1 + Q_2 + Q_3 = 1$. Figure 3-4 illustrates contours of

$$f(q_1, q_2, q_3) = \frac{\partial^3}{\partial q_1 \partial q_2 \partial q_3} F(q_1, q_2, q_3),$$

the joint pdf of $\mathbf{Q} = (Q_1, Q_2, Q_3)$, assuming shape parameters $a = 2$ and $b = 3$. Figure 3-5 then illustrates the approximation for the probability distribution of the true label, i.e., the solution to Eq. (3.2). The results indicate that the true label is of class C_1, C_2, and C_3 with probabilities 0.47, 0.16, and 0.37, respectively. We note that the mean values are $\mathbb{E}[Q_1] = 0.4$, $\mathbb{E}[Q_2] = 0.24$, and $\mathbb{E}[Q_3] = 0.36$, which are different. ◇

3.3. Measures of uncertainty

As mentioned in Section 2.1, we will limit our study to uncertainty due to limited training data and due to extrapolation error; these are sources 1(b) and 1(c) from Section 2.1. We introduce measures for these two sources of uncertainty in Sections 3.3.1 and 3.3.2.

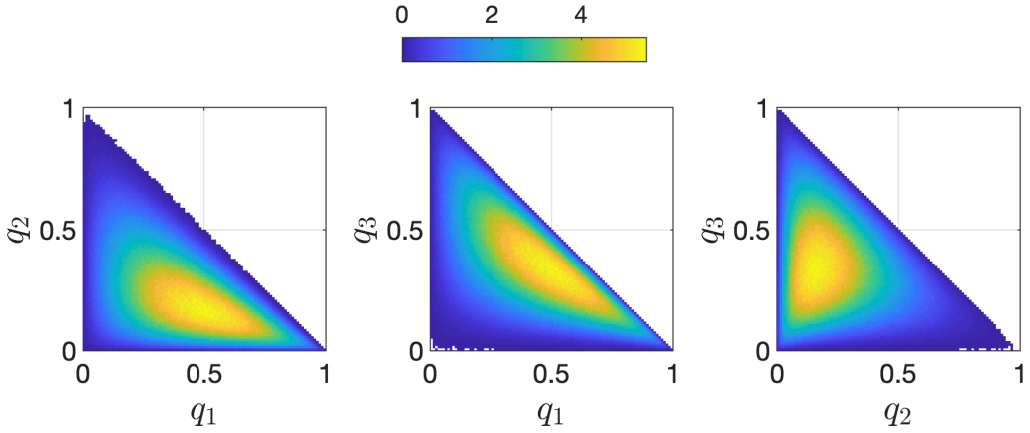

Figure 3-4. Contours of the joint pdf of $\mathbf{Q} = (Q_1, Q_2, Q_3)$ used in Example 3.2.3 with shape parameters $a = 2$ and $b = 3$.

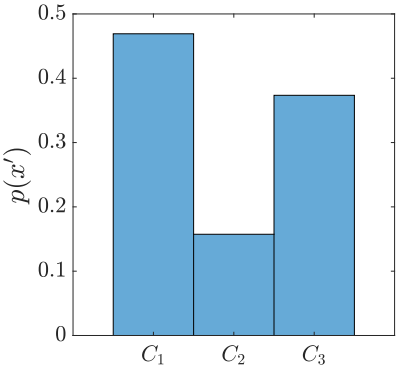

Figure 3-5. The approximation for $\mathbb{P}(c(\boldsymbol{x}') = C_i)$, $i = 1,2,3$ used in Example 3.2.3.

3.3.1. Minimum prediction deviation (MPD)

In practice, the joint cdf $F(\mathbf{q}) = \mathbb{P}(Q_1 \leq q_1, \ldots, Q_\kappa \leq q_\kappa)$ of $\mathbf{Q} = (Q_1, \ldots, Q_\kappa)$, where $Q_i = \hat{p}_i(\boldsymbol{x}'|\mathcal{T}) = \mathbb{P}(\hat{c}(\boldsymbol{x}') = C_i|\mathcal{T})$, is typically not known in closed-form. Instead, we are limited to a finite collection of independent random samples of $\mathbf{Q}$ which can be obtained, for example, by following procedure 1(b) for bootstrap sampling described in Section 3.1.

Accurate estimation of the full joint cdf of $\mathbf{Q}$ is difficult even when a large number of samples are available. We therefore propose an alternative measure of uncertainty for ML classification referred to as the minimum prediction deviation (MPD) [12], given by

$$\text{MPD}(\boldsymbol{x}'|\mathcal{T}) = \min_{i \in \{1,\ldots,\kappa\}} \left(\mathbb{E}[(1 - Q_i(\boldsymbol{x}'))^2] \right)^{1/2}. \tag{3.3}$$

$\text{MPD}(\boldsymbol{x}'|\mathcal{T})$ takes values in $[0,1]$. There are four special cases of interest:

1. The first case occurs when $Q_i(\boldsymbol{x}') = 1$ and $Q_j(\boldsymbol{x}') = 0$, $\forall j \neq i$; it is obvious that this case conveys the highest confidence in the classification of $\boldsymbol{x}'$ because the model predicts its label to be class i with probability one. It follows that $\text{MPD}(\boldsymbol{x}'|\mathcal{T}) = 0$ for this case.

2. The second case of interest occurs when $\mathbf{Q} = (1/\kappa, 1/\kappa, \ldots, 1/\kappa)$ with probability one. The joint distribution is equal to a Dirac delta with amplitude one located at the centroid of the $(\kappa - 1)$-simplex in $[0,1]^\kappa$. It follows that

$$\mathbb{E}[(1 - Q_i(\boldsymbol{x}'))^2] = \left(1 - \frac{1}{\kappa}\right)^2, \; i = 1, \ldots, \kappa,$$

   so that, by Eq. (3.3), $\text{MPD}(\boldsymbol{x}'|\mathcal{T}) = (\kappa - 1)/\kappa$. For the special case of $\kappa = 2$ classes, the joint distribution is $\Pr(Q_1 = 1/2, Q_2 = 1/2) = 1$. It should be clear that the ML classifier in this case is uncertain about the label for $\boldsymbol{x}'$ and is effectively flipping a coin on this instance.

3. The third case of interest occurs when $\mathbf{Q}$ is uniformly distributed over its support, i.e., the $(\kappa - 1)$-simplex in $[0,1]^\kappa$. This corresponds to a Dirichlet distribution with all concentration parameters set to unity (the "flat Dirichlet distribution"). It follows that the marginal distribution of any coordinate follows a beta distribution with shape parameters $(1, \kappa - 1)$. Further, we can show that

$$\mathbb{E}[(1 - Q_i(\boldsymbol{x}'))^2] = \frac{\kappa - 1}{\kappa + 1}, \; i = 1, \ldots, \kappa,$$

   so that, by Eq. (3.3), $\text{MPD}(\boldsymbol{x}'|\mathcal{T}) = \sqrt{(\kappa - 1)/(\kappa + 1)}$. For the special case of $\kappa = 2$, this corresponds to the case where Q_1 and Q_2 are both $\text{Unif}[0,1]$ random variables; for $\kappa > 2$, the marginal distributions are not uniformly distributed.

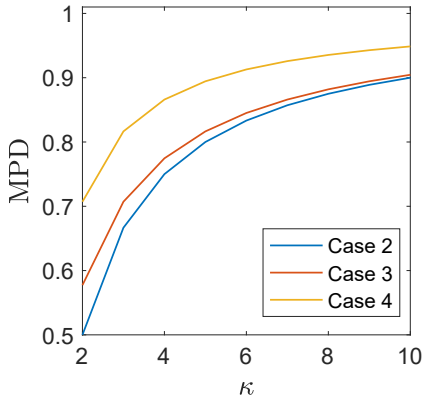

Figure 3-6. The MPD as a function of the number of classes κ for special cases 2, 3, and 4. We note that MPD $\to 1$ as $\kappa \to \infty$ for all cases.

4. The fourth case, corresponding to the highest level of uncertainty in the output of the ML model, occurs when

$$\mathbf{Q} = \begin{cases} (1,0,0,\ldots,0,0) & \text{with probability } 1/\kappa \\ (0,1,0,\ldots,0,0) & \text{with probability } 1/\kappa \\ \vdots & \vdots \\ (0,0,0,\ldots,0,1) & \text{with probability } 1/\kappa \end{cases}$$

   The joint distribution of $\mathbf{Q}$ is defined as collection of Dirac delta functions with amplitude $1/\kappa$ located at the each of the κ vertices of the $(\kappa-1)$-simplex in $[0,1]^\kappa$. It follows that

$$\mathbb{E}[(1-Q_i(\boldsymbol{x}'))^2] = (\kappa-1)/\kappa,\ i = 1,\ldots,\kappa,$$

   so that $\text{MPD}(\boldsymbol{x}'|\mathcal{T}) = \sqrt{(\kappa-1)/\kappa}$. For $\kappa = 2$ classes, this corresponds to the case where $\Pr(Q_1 = 0, Q_2 = 1) = 1/2$ and $\Pr(Q_1 = 1, Q_2 = 2) = 1/2$, that is, exactly half of the probability mass is concentrated at zero and the other half is concentrated at one.

We note that the MPD for these four cases satisfy the following

$$0 < \frac{\kappa-1}{\kappa} < \sqrt{\frac{\kappa-1}{\kappa+1}} < \sqrt{\frac{\kappa-1}{\kappa}} < 1, \forall \kappa > 1$$

so the cases are listed in order of increasing MPD. Figure 3-6 illustrates the MPD for cases 2, 3, and 4 as a function of the number of classes, κ. ◇

Example 3.3.1. To illustrate the concept of the MPD, let the output from the ML model $Q = \hat{p}(\boldsymbol{x}'|\mathcal{T})$ be a beta random variable with shape parameters a and b. The values for

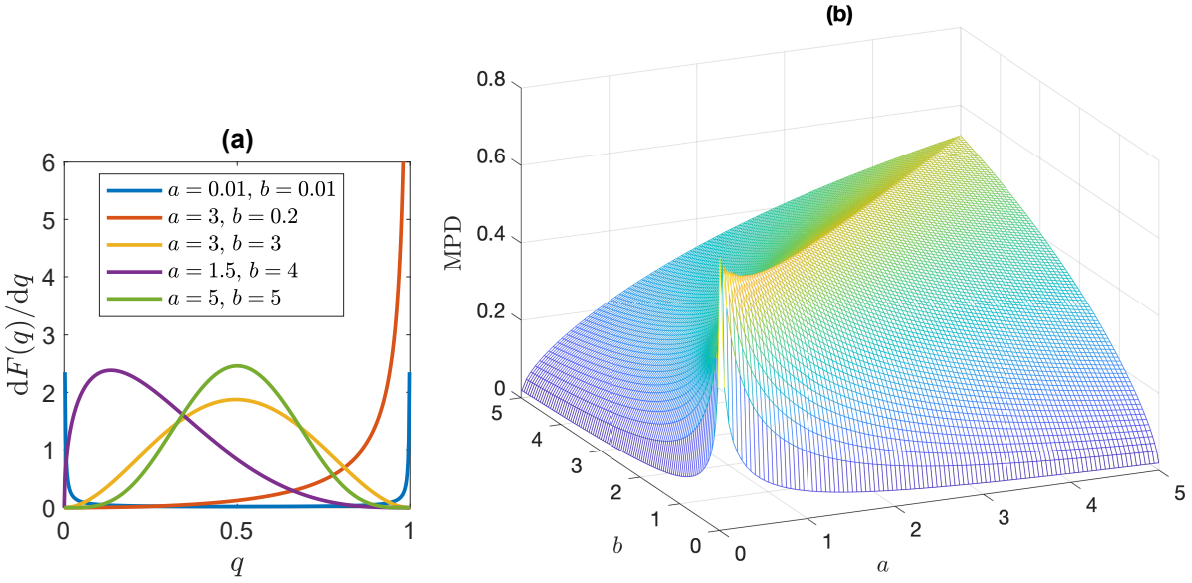

Figure 3-7. The MPD assuming $Q = \hat{p}(\boldsymbol{x}'|\mathcal{T})$ follows a beta distribution as described in Example 3.3.1. Panel (a) illustrates the pdf of Q for different values of the shape parameters, and panel (b) illustrates the MPD for $0 \leq a, b \leq 5$.

$\mathbb{E}[Q^2]$ and $\mathbb{E}[(1-Q)^2]$ can be obtained from the mean and variance of Q; it follows that the MPD for this case can be obtained in closed-form, i.e.,

$$\mathrm{MPD}(\boldsymbol{x}'|\mathcal{T}) = \min\left\{ \left(\frac{a\,b + a^2(a+b+1)}{(a+b)^2\,(a+b+1)} \right)^{1/2}, \left(\frac{b^3 + ab^2 + b^2 + ab}{(a+b)^2\,(a+b+1)} \right)^{1/2} \right\}.$$

Several cases of the MPD for the beta random variable are illustrated by Fig. 3-7(a). For example, with $a = b = 0.1$ (blue line), the MPD is 0.704. Uncertainty is high when $a = b = 0.1$ because the pdf for Q indicates equal and high confidence that $\boldsymbol{x}'$ is of either class. In contrast, the red curve ($a = 3$, $b = 0.2$) indicates a scenario with small uncertainty; as a grows large and b approaches zero (or vice-versa) we expect the MPD to approach zero.

Figure 3-7(b) illustrates the value of the MPD for the beta random variable as a function of $0 < a, b \leq 5$. As expected, the uncertainty is greatest when $a = b$ and a, b are both small; in this case MPD $\to 1/\sqrt{2}$. Further, as the difference $|a - b|$ grows large, the ML classifier starts to favor one class over the other, and MPD $\to 0$. $\diamond$

3.3.2. Local outlier factor (LOF)

To address extrapolation error, i.e., item 1(c) from Section 3.1, we apply the Local Outlier Factor (LOF) algorithm. The LOF algorithm [11] is an unsupervised anomaly detection method which computes the local density deviation of a given data point with respect to its neighbors. It considers as outliers the samples that have a substantially lower density than their neighbors.

The LOF score depends on the training set and is non-negative for any instance. Those instances $\boldsymbol{x}'$ that have LOF scores that are significantly larger than unity are labeled as outliers, that is, instance $\boldsymbol{x}'$ is deemed an outlier if $\text{LOF}(\boldsymbol{x}'|\mathcal{T}) \gg 1$.

Example 3.3.2. We now provide an example to illustrate the use of the LOF for outlier detection. The example dataset is constructed as follows:

- Let $G \sim N(0, 0.09)$ be a Gaussian random variable with zero mean and variance 0.09. We will build the nominal or "inlier" data from G.
- Next let $U \sim \text{Unif}[-4, 4]$ be a random variable distributed uniformly over $[-4, 4]$. We will build the anomalous or "outlier" data from U.
- We create a total of 220 instances $\boldsymbol{x}$, each with $d = 2$ features. The first 100 instances, denoted by $\boldsymbol{x}_{1:100}$, are defined by 100 independent samples of vector $(G + 2, G + 2)$. The next 100 instances, $\boldsymbol{x}_{101:200}$, are defined by 100 independent samples of vector $(G - 2, G - 2)$. The final 20 instances, $\boldsymbol{x}_{201:220}$, are defined by 20 independent samples of vector (U, U).
- The corresponding labels are $c(\boldsymbol{x}_{1:200}) = \texttt{inlier}$ and $c(\boldsymbol{x}_{201:220}) = \texttt{outlier}$. The dataset is illustrated by Fig. 3-8(a); the nominal/inlier and anomalous/outlier data points are designated by blue and red dots, respectively.

The LOF can be calculated for each point in the dataset; for calculations, we apply the `sklearn.neighbors.LocalOutlierFactor` algorithm with parameter `n_neighbors`=20. Figure 3-8(b) illustrates each of the 220 data points as circles with radii equal to the LOF score. Small circles are predicted to be inliers, and large circles are predicted to be outliers. The algorithm automatically determines a threshold such that any point with LOF score greater than the threshold is labeled an outlier. The color of the circle of the circle indicates whether the prediction was correct (green) or incorrect (red). For this example, there were 8 misclassifications: 5 inliers were incorrectly labeled as outliers, and 3 outliers were incorrectly labeled as inliers. ◇

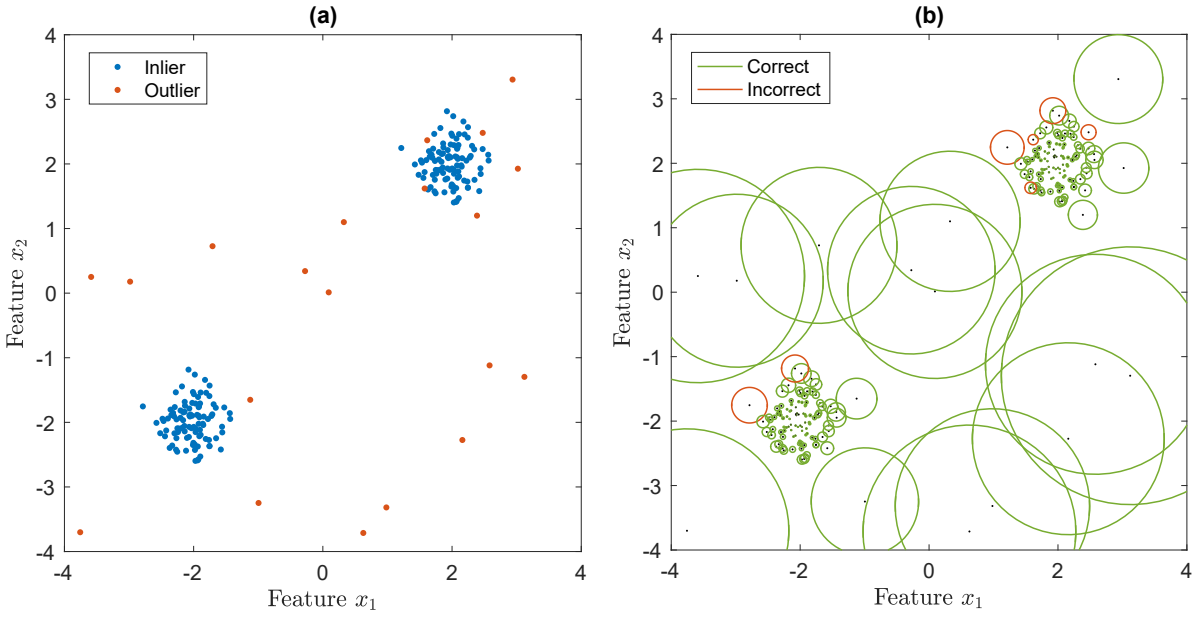

Figure 3-8. Predicting outliers using LOF. Panel (a) illustrates the ground truth: the dataset consists of 220 points, 20 of which are outliers. Panel (b) illustrates each data point as a circle with radius equal to the LOF score: 8 data points are misclassified by the algorithm (red circles).

3.4. Decisions under uncertainty

We introduced decision theory for ML in Section 2.1.1, but our formulation did not account for any sources of uncertainty in the model output. In this section, we extend the approach to account for uncertainty due to limited training data and extrapolation error.

There are two steps. Given a trained ML model and new instance $\boldsymbol{x}'$, we first apply the MPD and LOF measures defined in Sections 3.3.1 and 3.3.2 to establish whether or not the ML model should even be trusted to make a prediction on the label for $\boldsymbol{x}'$. To establish this trust, let $t_{\text{MPD}} \geq 0$ be a threshold parameter which quantifies the maximum degree of uncertainty (as measured by the MPD) that can be tolerated by the decision maker. Hence, if $\text{MPD}(\boldsymbol{x}'|\mathcal{T}) > t_{\text{MPD}}$, then we conclude that the ML model is incapable of making a reliable prediction on instance $\boldsymbol{x}'$ and should not be trusted for decision making.

Similarly, let $t_{\text{LOF}} \geq 1$ be a parameter that quantifies the maximum degree of extrapolation error that can be tolerated as measured by the LOF. If $\text{LOF}(\boldsymbol{x}'|\mathcal{T}) > t_{\text{LOF}}$, the ML model is deemed incapable of making a reliable prediction on instance $\boldsymbol{x}'$ and should not be trusted for decision making.

If both $\text{MPD}(\boldsymbol{x}'|\mathcal{T}) \leq t_{\text{MPD}}$ and $\text{LOF}(\boldsymbol{x}'|\mathcal{T}) \leq t_{\text{LOF}}$, we can establish trust in the ML model

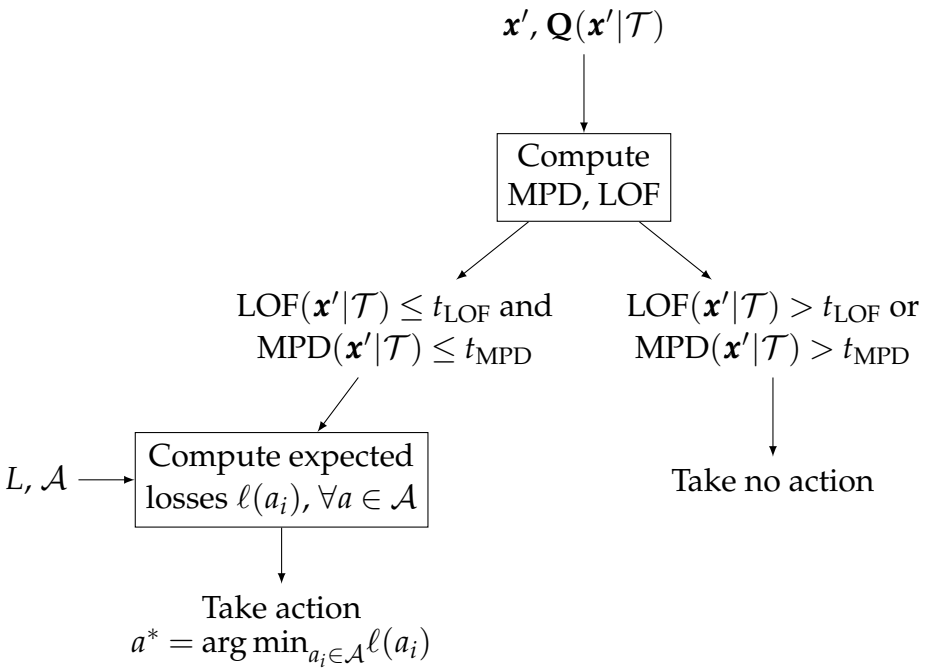

Figure 3-9. Decision process under uncertainty.

and proceed to step two. Based on the results from the Section 3.2, we can define the expected loss of action $a_i \in \mathcal{A}$ as

$$\ell(a_i) = \mathbb{E}[L(a_i, \mathcal{C})] = \sum_{j=1}^{\kappa} L(a_i, C_j) p_j(\boldsymbol{x}') \approx \sum_{j=1}^{\kappa} L(a_i, C_j) \mathbb{P}(Q_j > Q_k, \forall j \neq k), \tag{3.4}$$

where the loss function L is defined by Eq. (2.3) and probability $\mathbb{P}(Q_j > Q_k, \forall j \neq k)$ is defined by Eq. (3.2). A flowchart of the overall decision-making process is illustrated by Fig. 3-9.

REFERENCES

[1] Michael C. Darling, Richaerd V. Field Jr., Mark A. Smith, Jason W. Boada, Bickel J. Eric, Justin W. Doak, James M. Headen, Zachary J. Smith, David J., and David J. Stracuzzi. Decision science for machine learning (desciml). Technical Report SAND2022-xxxx, Sandia National Laboratories, September 2022.

[2] David J. Stracuzzi, Maximillian G. Chen, Michael C. Darling, Matthew G. Peterson, and Charles Vollmer. Uncertainty quantification for machine learning. Technical Report SAND2017-6776, Sandia National Laboratories, June 2017.

[3] David J. Stracuzzi, Michael C. Darling, , Matthew G. Peterson, and Maximillian G. Chen. Quantifying uncertainty to improve decision making in machine learning. Technical Report SAND2018-11166, Sandia National Laboratories, September 2018.

[4] Michael C. Darling and David J. Stracuzzi. Toward uncertainty quantification for supervised classification. Technical Report SAND2018-0032, Sandia National Laboratories, January 2018.

[5] David J. Stracuzzi, Michael C. Darling, Maximillian G. Chen, and Matthew G. Peterson. Data-driven uncertainty quantification for multisensor analytics. In *Ground/Air Multisensor Interoperability, Integration, and Networking for Persistent ISR IX*, volume 10635, page 106350N. International Society for Optics and Photonics, 2018.

[6] Charlie Vollmer, Matt Peterson, David J. Stracuzzi, and Maximillian G. Chen. Using data-driven uncertainty quantification to support decision making. In *Statistical Data Science*. World Scientific, 2017.

[7] P. Flach. *Machine Learning: The Art and Science of Algorithms that Make Sense of Data*. Cambridge University Press, 2012.

[8] H. Chernoff and L. E. Moses. *Elementary Decision Theory*. Dover Publications, Inc., New York, NY, 1959.

[9] J. von Neumann and O. Morgenstern. *Theory of Games and Economical Behavior*. Princeton University Press, Princeton, 1944.

[10] Eyke Hüllermeier and Willem Waegeman. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine Learning*, 110 (3):457–506, Mar 2021. ISSN 1573-0565. doi: 10.1007/s10994-021-05946-3. URL http://dx.doi.org/10.1007/s10994-021-05946-3.

[11] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. LOF: Identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, pages 93–104, 2000.

[12] Michael C Darling. *Using Uncertainty To Interpret Supervised Machine Learning Predictions*. PhD thesis, University of New Mexico, 2019.

DISTRIBUTION

Email—Internal

Name	Org.	Sandia Email Address
Technical Library	1911	sanddocs@sandia.gov

Sandia National Laboratories

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.