Calibrating Predictions to Decisions: A Novel Approach to Multi-Class Calibration

Anonymous Author(s)
Affiliation
Address
email

Abstract

When facing uncertainty, decision-makers want predictions they can trust. A machine learning provider can convey confidence to decision-makers by guaranteeing their predictions are distribution calibrated— amongst the inputs that receive a predicted vector of class probabilities q, the actual distribution over classes is given by q. For multi-class prediction problems, however, directly optimizing predictions under distribution calibration tends to be infeasible, requiring sample complexity that grows exponentially in the number of classes C. In this work, we introduce a new notion—*decision calibration*—that requires the predicted distribution and true distribution over classes to be "indistinguishable" to downstream decision-makers. This perspective gives a new characterization of distribution calibration: a predictor is distribution calibrated if and only if it is decision calibrated with respect to *all* decision-makers. Our main result shows that under a mild restriction, unlike distribution calibration, decision calibration is actually feasible. We design a recalibration algorithm that provably achieves decision calibration efficiently, provided that the decision-makers have a bounded number of actions (e.g., polynomial in C). We validate our recalibration algorithm empirically: compared to existing methods, decision calibration improves decision-making on skin lesion and ImageNet classification with modern neural network predictors.

1 Introduction

Machine learning predictions are increasingly employed by downstream decision makers who have little or no visibility on how the models were designed and trained. In high-stakes settings, such as healthcare applications, decision makers want predictions they can trust. For example, suppose a machine learning service offers a supervised learning model to healthcare providers that claims to predict the probability of various skin diseases, given an image of a lesion. Each healthcare provider may want assurance that the model's predictions lead to beneficial decisions, according to their own loss function. Concretely, the healthcare providers may reasonably worry that the model was trained using a loss function different than their own. This mismatch is often inevitable because the ML service may provide the same prediction model to many healthcare providers, which may have different treatment options available and loss functions. Even the same healthcare provider could have different loss functions throughout time, due to changes in treatment availability.

If predicted probabilities were perfect—exactly matching the corresponding empirical frequencies—the issue would not arise because they would lead to optimal decision making regardless of the loss function or task considered by downstream decision makers. In practice, however, predicted probabilities are never perfect. To address this, the healthcare providers may insist that the prediction function be at least *distribution calibrated*, requiring that amongst the inputs that receive predicted class probabilities q, the actual distribution over classes is q. This solves the trust issue because among

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

the patients who receive prediction q, the healthcare providers knows their true label distribution (which is also q) and hence know the true expected loss of a treatment on these patients. Unfortunately, to achieve distribution calibration, we need to reason about the set of individuals x who receive prediction q, for *every* possible predicted q. As the number of distinct predictions may naturally grow exponentially in the number of classes C, the amount of data needed to accurately certify distribution calibration tends to be prohibitive. Due to this statistical barrier, most work on obtaining calibrated multi-class predictions focuses on obtaining relaxed variants of calibration. These include *confidence calibration* (Guo et al., 2017), which calibrates predictions only over the most likely class, and *classwise calibration* (Kull et al., 2019), which calibrates predictions for each class marginally. While feasible, intuitively, these notions are significantly weaker than distribution calibration and do not address the trust issue highlighted above. Is there a calibration notion that addresses the issue of trust, but can also be verified and achieved efficiently? Our paper answers this question affirmatively.

Our Contributions. We introduce a new notion of calibration—*decision calibration*—where we take the perspective of potential decision-makers: the only differences in predictions that matter are those that could lead to different decisions. Inspired by Dwork et al. (2021), we formalize this intuition by requiring that predictions are "indistinguishable" from the true outcomes, according to a collection of decision-makers.

First, we show that all prior notions of calibration can be characterized as variants of decision calibration under different decision-makers. This framing explains the strengths and weakness of existing notions of calibration. For example, we show that a predictor is distribution calibrated if and only if it is decision calibrated with respect to *all* loss functions and decision rules. This characterization demonstrates why distribution calibration is so challenging: achieving distribution calibration requires simultaneously reasoning about all possible decision tasks.

Distribution calibration can guarantee decision calibration with respect to decision rules that choose between exponentially (in number of classes C) many actions. In practice, however, decision-makers typically choose from a bounded (or slowly-growing as a function of C) set of actions. Our main contribution is an algorithm that **guarantees decision calibration for such realistic decision-makers**. In particular, we give a sample-efficient algorithm that takes a pre-trained predictor and post-processes it to achieve decision calibration with respect to *all* decision-makers choosing from a bounded set of actions. Our recalibration procedure does not harm other performance metrics, and slightly improves accuracy and likelihood of the predictions. Empirically, we use our algorithm to recalibrate deep network predictors on two large scale datasets: skin lesion classification (HAM10000) and Imagenet. Our recalibration algorithm improves decision making, and allow for significantly more accurate loss estimation compared to existing recalibration methods.

2 Background

2.1 Setup and Notation

We consider the prediction problem with random variables X and Y. $X \in \mathcal{X}$ denotes the input features, and $Y \in \mathcal{Y}$ denotes the label. We focus on classification where $\mathcal{Y} = \{(1, 0, \cdots, 0), (0, 1, \cdots, 0), \cdots, (0, 0, \cdots, 1)\}$ where each $y \in \mathcal{Y}$ is a one-hot vector with $C \in \mathbb{N}$ classes. A probability prediction function is a map $\hat{p} : \mathcal{X} \to \Delta^C$ where Δ^C is the C-dimensional simplex. We use $p^* : \mathcal{X} \to \Delta^C$ to denote the true conditional probability, i.e. $p^*(x) = \mathbb{E}[Y \mid X = x]$.

2.2 Decision Making Tasks and Loss Functions

We formalize a decision making task as a utility maximization / loss minimization problem. The decision maker has some set of available actions $\mathcal{A}$ and a loss function $\ell : \mathcal{Y} \times \mathcal{A} \to \mathbb{R}$. In this paper we assume the loss function does not directly depend on the input features X. For notational simplicity we often refer to a (action set, loss function) pair $(\mathcal{A}, \ell)$ only by the loss function ℓ: the set of actions $\mathcal{A}$ is implicitly defined by the domain of ℓ. We denote the set of all possible loss functions as $\mathcal{L}_{\text{all}} = \{\ell : \mathcal{Y} \times \mathcal{A} \to \mathbb{R}\}$.

We treat all action sets $\mathcal{A}$ with the same cardinality as the same set — they are equivalent up to renaming the actions. A convenient way to think about this is that we only consider actions sets $\mathcal{A} \in \{[1], [2], \cdots, [K], \cdots, \mathbb{N}, \mathbb{R}\}$ where $[K] = \{1, \cdots, K\}$.

2.3 Bayes Decision

Given some predicted probability $\hat{p}(X)$ on Y, a decision maker selects an action in $\mathcal{A}$. We assume that the decision maker selects the action based on the predicted probability, that is we define a decision function as any map from the predicted probability to an action $\delta : \Delta^C \to \mathcal{A}$. and denote by $\mathbb{\Delta}_{\text{all}}$ as the set of all decision functions.

Typically a decision maker selects the action that minimizes the expected loss (under the predicted probability). This is formalized by the following definition

Definition 1 (Bayes Decision)**.** *Choose any $\ell \in \mathcal{L}_{\text{all}}$ with corresponding action space $\mathcal{A}$ and prediction $\hat{p}$, define the Bayes decision function as $\delta_\ell(\hat{p}(x)) = \arg\inf_{a \in \mathcal{A}} \mathbb{E}_{\hat{Y} \sim \hat{p}(x)}[\ell(\hat{Y}, a)]$. For any subset $\mathcal{L} \subset \mathcal{L}_{\text{all}}$ denote the set of all Bayes decision functions as $\mathbb{\Delta}_{\mathcal{L}} := \{\delta_\ell, \ell \in \mathcal{L}\}$.*

3 Calibration: A Decision Making Perspective

In our setup, the decision maker outsources the prediction task and uses a prediction function $\hat{p}$ provided by a third-party forecaster (e.g., an ML Prediction API). A decision maker would ideally want to train a prediction function $\hat{p}$ (or even more directly, a decision rule) to minimize the decision loss, however, this is often overly expensive, and decision makers often prefer to settle for off-the-shelf prediction functions due to resource constraints. A generic prediction function will have to be trained by a forecaster without knowledge of the exact loss ℓ downstream decision makers will use (or worse, will be used by multiple decision-makers with different loss functions). For example, prediction functions are often trained to minimize standard objectives such as L2 error or log likelihood.

This would not be an issue if the forecaster can perfectly learn the true distribution (i.e. $\hat{p}(X) = p^*(X)$ almost surely). Unfortunately, predictions are usually imperfect, and a prediction function trained to minimize L2 or log likelihood loss might or might not perform well on downstream tasks depending on the loss ℓ used. To mitigate concerns about the performance of the prediction function, the forecasters might offer (or decision makers might demand) performance guarantees applicable to decision makers with any loss function $\ell \in \mathcal{L}_{\text{all}}$. We also consider the case where the forecaster provides guarantees for decision makers whose loss functions come from some subset $\mathcal{L} \subset \mathcal{L}_{\text{all}}$.

3.1 Decision Calibration

First and foremost, a decision maker wants assurance that making decisions based on the prediction function $\hat{p}$ give low expected loss. Specifically, a decision maker with loss function ℓ naturally wants to use the Bayes decision rule δ_ℓ, so she wants δ_ℓ to incur a lower loss compared to alternative decision rules. Second, the decision maker wants to know how much loss is going to be incurred (before the actions are deployed and outcomes are revealed); the decision maker does not want to incur any additional loss in surprise that she has not prepared for.

To capture these desiderata we formalize a definition based on the following intuition: suppose a decision maker with some loss function ℓ considers a decision rule $\delta \in \mathbb{\Delta}_{\text{all}}$ (that may or may not be the Bayes decision rule), the decision maker should be able to correctly compute the expected loss of using δ to make decisions.

Definition 2 (Decision Calibration)**.** *For any set of loss functions $\mathcal{L} \subset \mathcal{L}_{\text{all}}$ and set of decision rules $\mathbb{\Delta} \subset \mathbb{\Delta}_{\text{all}} := \{\Delta^C \to \mathcal{A}\}$, we say that a prediction $\hat{p}$ is $(\mathcal{L}; \mathbb{\Delta})$-decision calibrated (with respect to p^*) if $\forall \ell \in \mathcal{L}$ and $\delta \in \mathbb{\Delta}$ with the same action space $\mathcal{A}$* [1]

$$\mathbb{E}_X \mathbb{E}_{\hat{Y} \sim \hat{p}(X)}[\ell(\hat{Y}, \delta(\hat{p}(X)))] = \mathbb{E}_X \mathbb{E}_{Y \sim p^*(X)}[\ell(Y, \delta(\hat{p}(X)))] \tag{1}$$

In particular, we say $\hat{p}$ is $\mathcal{L}$-decision calibrated if it is $(\mathcal{L}; \mathbb{\Delta}_{\mathcal{L}})$-decision calibrated.

The left hand side of Eq.(1) is the "simulated" loss where the outcome $\hat{Y}$ is hypothetically drawn from the predicted distribution. The decision maker can compute this just by knowing the input features X and without knowing the outcome Y. The right hand side of Eq.(1) is the true loss that the decision maker incurs in reality if she uses the decision rule δ. Note that Eq.(1) should not be mis-interpreted

[1] We require ℓ and δ to have the same action space $\mathcal{A}$ for type check reasons. Eq.(1) only has meaning if the loss ℓ and decision rule $\mathbb{\Delta}$ are associated with the same action space $\mathcal{A}$.

Existing Calibration Definitions	Associated Loss Functions
Confidence Calibration (Guo et al., 2017) $\Pr[Y = \arg\max \hat{p}(X) \mid \max \hat{p}(X) = \beta] = \beta$ $\forall \beta \in [0, 1]$	$\mathcal{L}_r := \left\{ \begin{array}{l} \ell(y, a) = \mathbb{I}(y \neq a \cap a \neq \perp) + \beta \mathbb{I}(a = \perp) : \\ a \in \mathcal{Y} \cup \{\perp\}, \forall \beta \in [0, 1] \end{array} \right\}$
Classwise Calibration (Kull et al., 2019) $\mathbb{E}[Y_c \mid \hat{p}_c(X) = \beta] = \beta, \forall c \in [C], \forall \beta \in [0, 1]$	$\mathcal{L}_{cr} := \left\{ \begin{array}{r} \ell(y, a) = \mathbb{I}(a = \perp) + \beta_1 \mathbb{I}(y = c, a = T) \\ +\beta_2 \mathbb{I}(y \neq c, a = F) : \\ a \in \{T, F, \perp\}, \forall \beta_1, \beta_2 \in \mathbb{R}, c \in [C] \end{array} \right\}$
Distribution Calibration (Kull & Flach, 2015) $\mathbb{E}[Y \mid \hat{p}(X) = q] = q, \forall q \in \Delta^C$	$\mathcal{L}_{\text{all}} = \{\ell : \mathcal{Y} \times \mathcal{A} \mapsto \mathbb{R}\}$

Table 1: A prediction function $\hat{p}$ satisfies the calibration definitions on the left if and only if it satisfies $\mathcal{L}$-decision calibration for the loss function families on the right (Theorem 1).

as guarantees about individual decisions; Eq.(1) only looks at the average loss when X, Y is a random draw from the population. Individual guarantees usually impossible (Zhao & Ermon, 2021).

We define the special notion of $\mathcal{L}$-decision calibrated because given a set of loss functions $\mathcal{L}$, we are often only interested in the associated Bayes decision rules $\mathbb{\Delta}_{\mathcal{L}}$, i.e. the set of decision rules that are optimal under *some* loss function. For the rest of the paper we focus on $\mathcal{L}$-decision calibration for simplicity. $\mathcal{L}$-decision calibration can capture the desiderata we discussed above

Proposition 1. *If a prediction function $\hat{p}$ is $\mathcal{L}$-decision calibrated, then it satisfies*

$$\forall \delta' \in \mathbb{\Delta}_{\mathcal{L}}, \mathbb{E}_X \mathbb{E}_{Y \sim p^*(X)}[\ell(Y, \delta_\ell(\hat{p}(X)))] \leq \mathbb{E}_X \mathbb{E}_{Y \sim p^*(X)}[\ell(Y, \delta'(\hat{p}(X)))] \quad \textit{(No regret)}$$

$$\mathbb{E}_X \mathbb{E}_{\hat{Y} \sim \hat{p}(X)}[\ell(\hat{Y}, \delta_\ell(\hat{p}(X)))] = \mathbb{E}_X \mathbb{E}_{Y \sim p^*(X)}[\ell(Y, \delta_\ell(\hat{p}(X)))] \quad \textit{(Accurate loss estimation)}$$

No regret states that the Bayes decision rule δ_ℓ is not worse than any alternative decision rule $\delta' \in \mathbb{\Delta}_{\mathcal{L}}$. Note we do not guarantee no regret with respect to any decision rule in $\mathbb{\Delta}_{\text{all}}$. This is going to be a mild restriction because we will achieve decision calibration with respect to a very large class of loss functions $\mathcal{L}$ (any loss function with a bounded number of actions) such that $\mathbb{\Delta}_{\mathcal{L}}$ will consist of almost all realistic decision rules.

Accurate loss estimation states that for the Bayes decision rule δ_ℓ, the simulated loss on the left hand side (which the decision maker can compute before the outcomes are revealed) equals the true loss on the right hand side. This ensures that the decision maker knows the expected loss it's going to incur and can prepare for it.

In practice, the forecaster chooses some set $\mathcal{L}$ to achieve $\mathcal{L}$-decision calibration, and advertise it to decision makers. A decision makers can then check whether their loss function ℓ belongs to the advertised set $\mathcal{L}$. If it does, the decision maker should be confident that the Bayes decision rule δ_ℓ has low loss compared to alternatives in $\mathbb{\Delta}_{\mathcal{L}}$, and they can know in advance the loss that will be incurred.

3.2 Decision Calibration Generalizes Existing Notions of Calibration

This section connects decision calibration with existing notions of calibration in Table 1. Decision calibration provides a unified view of existing notions of calibration as the following theorem shows.

Theorem 1 (Calibration Equivalence)**.** *For any true distribution p^*, and for the loss function sets $\mathcal{L}_r, \mathcal{L}_{cr}$ defined in Table 1, a prediction function $\hat{p}$ is*

- *confidence calibrated iff it is $\mathcal{L}_r$-decision calibrated.*
- *classwise calibrated iff it is $\mathcal{L}_{cr}$-decision calibrated.*
- *distribution calibrated iff it is $\mathcal{L}_{all}$-decision calibrated.*

For proof of this theorem see Appendix D. In Theorem 1 and Table 1, confidence and classwise calibration are weak notions of calibration; correspondingly the loss function families $\mathcal{L}_r$ and $\mathcal{L}_{\text{cr}}$ are also very restricted. For example, confidence calibration only requires that the top-1 accuracy $\Pr[Y = \arg\max \hat{p}(X)]$ matches the predicted top-1 probability $\max \hat{p}(X)$. Correspondingly, $\mathcal{L}_r$ consists of loss functions for the refrained prediction task: a decision maker chooses between reporting a class label, or reporting "I don't know," denoted $\perp$. If the decision maker reports a class label, she incurs a loss of 1 if the reported class label is incorrect; if the decision maker reports "I don't know" then she incurs a loss of $\beta < 1$. Even though such decision tasks are useful, they only account for a tiny subset of all possible tasks that are interesting to decision makers.

On the other hand, distribution calibration (i.e. $\mathbb{E}[Y \mid \hat{p}(X) = q] = q, \forall q \in \Delta^C$) is equivalent to $\mathcal{L}_{\text{all}}$-decision calibration. This means that decision makers with any loss functions are guaranteed the properties of no regret and accurate loss estimation as in Proposition 1. Unfortunately, distribution calibration is very challenging to verify or achieve. To understand the challenges, consider certifying whether a given predictor $\hat{p}$ is distribution calibrated. Because we need to reason about the conditional distribution $\mathbb{E}[Y \mid \hat{p}(X) = q]$ for every q that $\hat{p}$ can predict (i.e. the support of $\hat{p}$), the necessary sample complexity grows linearly in the support of $\hat{p}$. Of course, for a trivial predictors that map all inputs x to the same prediction q_0 (i.e. $\hat{p}(x) = q_0, \forall x \in \mathcal{X}$) distribution calibration is easy to certify (Widmann et al., 2019), but such predictors have no practical use, and informative predictors naturally have exponentially large (in the number of classes C) support.

Our characterization of distribution calibration further sheds light on why it is so difficult to achieve. $\mathcal{L}_{\text{all}}$ consists of *all* loss function, including all loss functions $\ell : \mathcal{Y} \times \mathcal{A} \to \mathbb{R}$ whose action space $\mathcal{A}$ contains exponentially many elements (e.g. $|\mathcal{A}| = 2^C$). The corresponding decision rules $\delta \in \mathbb{\Delta}_{\mathcal{L}_{\text{all}}} : \Delta^C \to \mathcal{A}$ may also map Δ^C to exponentially many possible values. Intuitively, a predictor that guarantees $\mathcal{L}_{\text{all}}$-decision calibration guarantees accurate loss estimation even when the losses and decisions are specified separately for each $q \in \Delta^C$ (just as distribution calibration conditions on $\hat{p}(X) = q$ for each $q \in \Delta^C$). Enforcing Definition 2 for such complex loss functions and decision rules is naturally difficult. But we argue that it may also be *unnecessary* in many contexts, as realistic decision makers often choose from a bounded set of discrete actions.

4 Achieving Decision Calibration with PAC Guarantees

In this section, we consider obtaining decision calibration for all loss functions defined over a bounded number of actions $K \in \mathbb{N}$. Formally, let $\mathcal{L}^K$ be the set of all loss functions with K possible actions, i.e. $\mathcal{L}^K = \{\ell : \mathcal{Y} \times \mathcal{A} \to \mathbb{R}, |\mathcal{A}| = K\} \subset \mathcal{L}_{\text{all}}$. In practice, decision-makers are unlikely to have an unbounded number of actions, so for most realistic settings, the relevant losses will be in $\mathcal{L}^K$ for some reasonable $K \in \mathbb{N}$. In the remainder of this section, we demonstrate an efficient strategy for obtaining approximately $\mathcal{L}^K$-decision calibrated predictors. In Appendix B we also explore the theoretical connection between $\mathcal{L}^K$-decision calibration and distribution calibration to show that $\mathcal{L}^K$-decision calibration improves upon distribution calibration in almost every way.

4.1 Approximate $\mathcal{L}^K$ Decision Calibration is Verifiable and Achievable

Decision calibration in Definition 2 usually cannot be achieved perfectly. The definition has to be relaxed to allow for statistical and numerical errors. To meaningfully define approximate calibration we must assume that the loss functions are bounded, i.e. no outcome $y \in \mathcal{Y}$ and action $a \in \mathcal{A}$ can incur an infinite loss. In particular, we bound ℓ by its 2-norm $\max_a \|\ell(\cdot, a)\|_2 := \max_a \sqrt{\sum_{y \in \mathcal{Y}} \ell(y, a)^2}$. [2]

Now we can proceed to define approximate decision calibration. In particular, we compare the difference between the two sides in Eq.(1) of Definition 2 with the maximum size of the loss function.

Definition 3 (Approximate decision calibration)**.** *A prediction function $\hat{p}$ is $(\mathcal{L}, \epsilon)$-decision calibrated (with respect to p^*) if $\forall \ell \in \mathcal{L}$ and $\delta \in \mathbb{\Delta}_{\mathcal{L}}$*

$$\left| \mathbb{E}[\ell(\hat{Y}, \delta(\hat{p}(X)))] - \mathbb{E}[\ell(Y, \delta(\hat{p}(X)))] \right| \leq \epsilon \sup_{a \in \mathcal{A}} \|\ell(\cdot, a)\|_2 \tag{2}$$

Definition 3 is a relaxation of Definition 2: if a prediction function is $(\mathcal{L}, 0)$-decision calibrated, then it is $\mathcal{L}$-decision calibrated (Definition 2).

The main observation in our paper is that for the set of loss functions with K actions, $(\mathcal{L}^K, \epsilon)$-decision calibration can be verified and achieved with polynomially many samples.

Theorem 2 (Main Theorem, informal)**.** *There is an algorithm, such that for any predictor $\hat{p}$ and given polynomially (in $K, C, 1/\epsilon$) many samples, can with high probability*

1. *correctly decide if $\hat{p}$ satisfies $(\mathcal{L}^K, \epsilon)$-decision calibration*

[2] The choice of 2-norm is for convenience. All p-norms are equivalent up to a multiplicative factor polynomial in C, so our main theorem (Theorem 2) still hold for any p-norms up to the polynomial factor.

2. *output a new predictor $\hat{p}'$ that satisfies $(\mathcal{L}^K, \epsilon)$-decision calibration without degrading the original predictions (in terms of L_2 error).*

As with distribution calibration, trivial predictors $\hat{p}(x) \equiv \mathbb{E}[Y], \forall x$ satisfy $(\mathcal{L}^K, 0)$-decision calibration. To maintain a meaningful prediction function we also require "sharpness" which we measure by L_2 error. We guarantee that the L_2 error of $\hat{p}'$ can only *improve* under post-processing; that is, $\mathbb{E}[\|\hat{p}'(X) - Y\|_2^2] \leq \mathbb{E}[\|\hat{p}(X) - Y\|_2^2]$. In fact, the algorithm works by iteratively updating the predictions to make progress in L_2. For rest of this section, we propose concrete algorithms that satisfy Theorem 2.

4.2 Verification of Decision Calibration

This section focuses on the first part of Theorem 2 where we certify $(\mathcal{L}^K, \epsilon)$-decision calibration. A naive approach would use samples to directly estimate

$$\sup_{\delta \in \mathbb{\Delta}_{\mathcal{L}^K}} \sup_{\ell \in \mathcal{L}^K} \left| \mathbb{E}[\ell(\hat{Y}, \delta(\hat{p}(X)))] - \mathbb{E}[\ell(Y, \delta(\hat{p}(X)))] \right| / \left(\sup_{a \in \mathcal{A}} \|\ell(\cdot, a)\|_2 \right) \tag{3}$$

and compare it with ϵ. However, the complexity of this optimization problem poses challenges to analysis. We will make several observations to transform this complex optimization problem to a simple problem that resembles linear classification.

Observation 1. The first observation is that we do not have to take the supremum over $\mathcal{L}^K$ because for any choice of $\delta \in \mathbb{\Delta}$ by simple calculations (details in the Appendix) we have

$$\sup_{\ell \in \mathcal{L}^K} \frac{\left| \mathbb{E}[\ell(\hat{Y}, \delta(\hat{p}(X)))] - \mathbb{E}[\ell(Y, \delta(\hat{p}(X)))] \right|}{\sup_{a \in \mathcal{A}} \|\ell(\cdot, a)\|_2} = \sum_{a=1}^{K} \left\| \mathbb{E}[(\hat{Y} - Y)\mathbb{I}(\delta(\hat{p}(X)) = a)] \right\|_2 \tag{4}$$

Intuitively on the right hand side, δ partitions the probabilities Δ^C based on the optimal decision $\Delta_1 := \{q \in \Delta^C, \mathbb{I}(\delta(q) = 1)\}, \cdots, \Delta_K := \{q \in \Delta^C, \mathbb{I}(\delta(q) = K)\}$. For each partition Δ_k we measure the difference between predicted label and true label *on average*, i.e. we compute $\mathbb{E}[(\hat{Y} - Y)\mathbb{I}(\hat{p}(X) \in \Delta_k)]$.

Observation 2. We observe that the partitions of Δ^C are defined by linear classification boundaries. Formally, we introduce a new notation for the linear multi-class classification functions

$$B^K = \left\{ b_w \mid \forall w \in \mathbb{R}^{K \times C} \right\} \qquad \text{where } b_w(q, a) = \mathbb{I}(a = \arg \sup_{a' \in [K]} \langle q, w_{a'} \rangle) \tag{5}$$

Note that this new classification task is a tool to aid in understanding decision calibration, and bears no relationship with the original prediction task (predicting Y from X). Intuitively w is the weights of a linear classifier; given input features $q \in \Delta^C$ and a candidate class a, b_w outputs an indicator: $b_w(q, a) = 1$ if the optimal decision of q is equal to a and 0 otherwise.

The following equality draws the connection between Eq.(4) and linear classification. The proof is simply a translation from the original notations to the new notations.

$$\sup_{\delta \in \mathbb{\Delta}_{\mathcal{L}^K}} \sum_{a=1}^{K} \left\| \mathbb{E}[(\hat{Y} - Y)\mathbb{I}(\delta(\hat{p}(X))) = a)] \right\|_2 = \sup_{b \in B^K} \sum_{a=1}^{K} \left\| \mathbb{E}[(\hat{Y} - Y) b(\hat{p}(X), a)] \right\|_2 \tag{6}$$

The final outcome of our derivations is the following proposition (Proof in Appendix D.2)

Proposition 2. *$\hat{p}$ satisfies $(\mathcal{L}^K, \epsilon)$-decision calibration if and only if*

$$\sup_{b \in B^K} \sum_{a=1}^{K} \left\| \mathbb{E}[(\hat{Y} - Y) b(\hat{p}(X), a)] \right\|_2 \leq \epsilon \tag{7}$$

In words, $\hat{p}$ satisfies decision calibration if and only if there is no linear classification function that can partition Δ^C into K parts, such that the average difference $\hat{Y} - Y$ (or equivalently $\hat{p}(X) - p^*(X)$) in each partition is large. Theorem 2.1 follows naturally because B^K has low Radamacher complexity,

so the left hand side of Eq.(7) can be accurately estimated with polynomially many samples. For a formal statement and proof see Appendix D.2.

The remaining problem is computation. With unlimited compute, we can upper bound Eq.(7) by brute force search over B^K; in practice, we use a surrogate objective optimizable with gradient descent. This is the topic of Section 4.4.

4.3 Recalibration Algorithm

This section discusses the second part of Theorem 2 where we design a post-processing recalibration algorithm. The algorithm is based on the following intuition, inspired by (Hébert-Johnson et al., 2018): given a predictor $\hat{p}$ we find the worst $b \in B^K$ that violates Eq.(7) (line 3 of Algorithm 1); then, we make an update to $\hat{p}$ to minimize the violation of Eq.(7) for the worst b (line 4,5 of Algorithm 1). This process is be repeated until we get a (B^K, ϵ)-decision calibrated prediction (line 2). The sketch of the algorithm is shown in Algorithm 1 and the detailed algorithm is in the Appendix.

Algorithm 1: Recalibration algorithm to achieve $\mathcal{L}^K$ decision calibration.

1 Input current prediction function $\hat{p}$, tolerance ϵ. Initialize $\hat{p}^{(0)} = \hat{p}$;
2 **for** $t = 1, 2, \cdots, T$ *until output* $\hat{p}^{(T)}$ *when it satisfies Eq.(7)* **do**
3 Find $b \in B^K$ that maximizes $\sum_{a=1}^{K} \left\| \mathbb{E}[(Y - \hat{p}^{(t-1)}(X))b(\hat{p}^{(t-1)}(X), a) \right\|_2$;
4 For each a compute the adjustment $d_a = \mathbb{E}[(Y - \hat{p}^{(t-1)}(X))b(\hat{p}^{(t-1)}(X), a)]$;
5 Set $\hat{p}^{(t)} : x \mapsto \pi\left(\hat{p}^{(t-1)}(x) + \sum_{a=1}^{K} b(\hat{p}^{(t-1)}(x), a) \cdot d_a\right)$, π is normalizing projection ;
6 **end**

Given a dataset with N samples, the expectations in Algorithm 1 are replaced with empirical averages. The following theorem demonstrates that Algorithm 1 satisfies the conditions stated in Theorem 2.

Theorem 2.2. *Given any input $\hat{p}$ and tolerance ϵ, Algorithm 1 terminates in $O(1/\epsilon^2)$ iterations. Given $O(poly(K, C, \log(1/\delta)))$ samples, with $1 - \delta$ probability Algorithm 1 outputs a $(\mathcal{L}^K, \epsilon)$-decision calibrated prediction function $\hat{p}'$ that satisfies $\mathbb{E}[\|\hat{p}'(X) - Y\|_2^2] \leq \mathbb{E}[\|\hat{p}(X) - Y\|_2^2]$.*

4.4 Relaxation of Decision Calibration for Computational Efficiency

We complete the discussion by addressing the open computational question. Directly optimizing over B^K is difficult, so we instead define the softmax relaxation.

$$\bar{B}^K = \left\{ \bar{b}_w \mid \forall w \in \mathbb{R}^{K \times C} \right\} \qquad \text{where } \bar{b}_w(q, a) = \frac{e^{\langle q, w_a \rangle}}{\sum_{a'} e^{\langle q, w_{a'} \rangle}}$$

The key motivation behind this relaxation is that $\bar{b}_w \in \bar{B}^K$ is now is differentiable in w, so we can optimize over $\bar{B}^K$ using gradient descent. Correspondingly some technical details in Algorithm 1 change to accommodate soft partitions; we address these modifications in Appendix A. Importantly, even when we optimize over $\bar{B}^K$, we still obtain our approximate decision calibration goal.

Proposition 3. *A prediction function $\hat{p}$ is $(\mathcal{L}^K, \epsilon)$-decision calibrated if it satisfies*

$$\sup_{\bar{b} \in \bar{B}^K} \sum_{a=1}^{K} \left\| \mathbb{E}[(\hat{Y} - Y)\bar{b}(\hat{p}(X), a)] \right\|_2 \leq \epsilon \tag{8}$$

5 Empirical Evaluation

5.1 Skin Legion Classification

This experiment materializes our motivating example in the introduction. We aim to show on a real medical prediction dataset, our recalibration algorithm improves both the decision loss and reduces the decision loss estimation error. For the estimation error, as in Definition 3 for any loss function ℓ and corresponding Bayes decision rule δ_ℓ we measure

$$\text{loss gap} := \left| \mathbb{E}[\ell(\hat{Y}, \delta_\ell(\hat{p}(X)))] - \mathbb{E}[\ell(Y, \delta_\ell(\hat{p}(X)))] \right| / \sup_{a \in \mathcal{A}} \|\ell(\cdot, a)\|_2$$

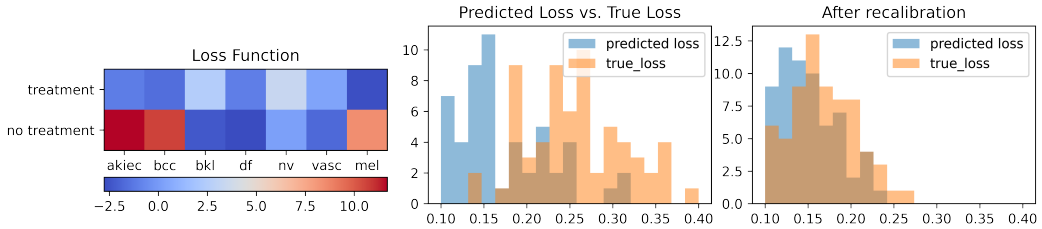

Figure 1: An illustrative example of accurate loss estimation on the HAM10000 dataset. The predictor predicts the probability of 7 skin illness categories (akiec, bcc, ..., mel); the hospital decides between two actions (treatment vs. no treatment). **Left** An example loss function (details not important). Blue indicates low loss and red indicates high loss. For the malignant conditions such as 'akiec' and 'mel', no treatment has high loss (red); for the benign conditions such as 'nv', (unnecessary) treatment has moderate loss. **Middle** Histogram of predicted loss vs. the true loss. The predicted loss is the loss the hospital expected to incur assuming the predicted probabilities are correct under the Bayes decision rule.The true loss is the loss the hospital actually incurs (after ground-truth outcomes are revealed). We plot the histogram from random train/test splits of the dataset, and observe that the true loss is generally much greater than the predicted loss. Because the hospital might incur loss it didn't expect or prepare for, the hospital cannot trust the prediction function. **Right** After applying our recalibration algorithm, the true loss almost matches the predicted loss.

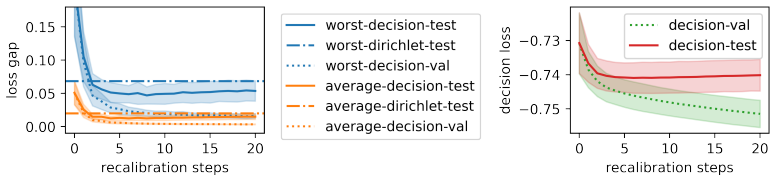

Figure 2: Calibration improves decision loss and its estimation on the HAM10000 skin legion classification dataset. **Left** The gap between the predicted decision loss and the true decision loss in Eq.(9) on a set of randomly sampled loss functions. We plot both the average gap (orange) and the worst gap (blue) out of all the loss functions. The dotted lines are the validation set performance, and solid lines are the test set performance. With more recalibration steps in Algorithm 1, both the average gap and the worst gap improves. The improvements are greater than Dirichlet recalibration (dashed line) **Right** The decision loss (averaged across all random loss functions). With more recalibration steps the decision loss also improves slightly.

In addition to the loss function in Figure 1 (which is motivated by medical domain knowledge), we also consider a set of 500 random loss functions where for each $y \in \mathcal{Y}, a \in \mathcal{A}$, $\ell(y, a) \sim \text{Normal}(0, 1)$, and report both the average loss gap and the maximum loss gap across the loss functions.

Setup We use the HAM10000 dataset (Tschandl et al., 2018). We partition the dataset into train/validation/test sets, where approximately 15% of the data are used for validation, while 10% are used for the test set. We use the train set to learn the baseline classifier $\hat{p}$, validation set to recalibrate, and the test set to measure final performance. For modeling we use the densenet-121 architecture (Huang et al., 2017), which achieves around 90% accuracy.

Methods For our method we use Algorithm 2 in Appendix A (which is a small extension of Algorithm 1 explained in Section 4.4). We compare with temperature scaling (Guo et al., 2017) and Dirichlet calibration (Kull et al., 2019). We observe that all recalibration methods (including ours) work better if we first apply temperature scaling, hence we first apply it in all experiments. For example, in Figure 2 temperature scaling corresponds to 0 decision recalibration steps.

Results The results are shown in Figure 2. For these experiments we set the number of actions $K = 3$. For other choices we obtain qualitatively similar results in Appendix C. The main observation is that decision recalibration improves the loss gap in Eq.(9) and slightly improves the decision loss. Our recalibration algorithm converges rather quickly (in about 5 steps). We also observe that our recalibration algorithm slightly improves top-1 accuracy (the average improvement is $0.40 \pm 0.08\%$) and L2 loss (the average decrease is 0.010 ± 0.001) on the test set.

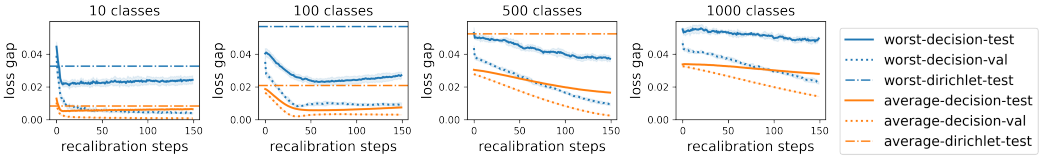

Figure 3: Calibration improves decision loss estimation on Imagenet-1000. The meaning of the plot is identical to Figure 2. From left to right we vary the number of classification categories C from 10 to 1000. Our algorithm reduces the gap between predicted loss and true loss in Eq.(9) even up to 1000 classes, although more classes require more iterations and are more prone to over-fitting. Dirichlet calibration is less scalable than our method. For example, with 1000 classes the loss gap of Dirichlet calibration is off the chart.

5.2 Imagenet Classification

We stress test our algorithm on Imagenet. The aim is to show that even with deeply optimized classifiers (such as inception-v3 and resnet) that are tailor made for the Imagenet benchmark and a large number of classes (1000 classes), our recalibration algorithm can improve the loss gap in Eq.(9).

Setup The setup and baselines are identical to the HAM10000 experiment with two differences: we use pretrained models provided by pytorch, and among the 50000 standard validation samples, we use 40000 samples for recalibration and 10000 samples for testing. Similar to the previous experiment, we randomly generate a set of 500 loss functions from normal distributions.

Results The results are shown in Figure 3 with additional plots in Appendix C. Decision calibration can generalize to a larger number of classes, and still provides some (albeit smaller) benefits with 1000 classes. Recalibration does not hurt accuracy and L2 error, as we observe that both improve by a modest amount (on average by 0.30% and 0.00173 respectively). We contrast decision calibration with Dirichlet calibration (Kull et al., 2019). Dirichlet calibration also reduces the loss gap when the number of classes is small (e.g. 10 classes), but is less scaleble than decision recalibration. With more classes its performance degrades much more than decision calibration.

6 Related Work

Calibration Early calibration research focus on binary classification (Brier, 1950; Dawid, 1984). For multiclass classification, the strongest definition is distribution (strong) calibration (Kull & Flach, 2015; Song et al., 2019) but is hindered by sample complexity. Weaker notions such as confidence (weak) calibration (Platt et al., 1999; Guo et al., 2017), class-wise calibration (Kull et al., 2019) or average calibration average calibration (Kuleshov et al., 2018) are more commonly used in practice. To unify these notions, (Widmann et al., 2019) proposes $\mathcal{F}$-calibration but lacks detailed guidance on which notions to use.

Individual calibration Our paper discusses the *average* decision loss over the population X. A stronger requirement is to guarantee the loss for each individual decision. Usually individual guarantees are near-impossible (Barber et al., 2019) and are only achievable with hedging (Zhao & Ermon, 2021) or randomization (Zhao et al., 2020).

Multi-calibration and Outcome Indistinguishability. Calibration have been the focus of many works on fairness, starting with (Kleinberg et al., 2016; Pleiss et al., 2017). Multi-calibration has emerged as a noteworthy notion of fairness (Hébert-Johnson et al., 2018; Kim et al., 2019; Dwork et al., 2019; Shabat et al., 2020; Jung et al., 2020; Dwork et al., 2021) because it goes beyond "protected" groups, and guarantees calibration for any group that is identifiable within some computational bound. Recently, (Dwork et al., 2021) generalizes multicalibration to outcome indistinguishability (OI). Decision calibration is a special form of OI.

7 Conclusion

We discussed why calibration is important for decision makers, and show that calibration typically implies two desiderata: no regret and accurate loss estimation. We showed that it is possible to achieve these desiderata for realistic decision makers choosing between a bounded number of actions.

References

Rina Foygel Barber, Emmanuel J Candes, Aaditya Ramdas, and Ryan J Tibshirani. The limits of distribution-free conditional predictive inference. *arXiv preprint arXiv:1903.04684*, 2019.

Glenn W Brier. Verification of forecasts expressed in terms of probability. *Monthly weather review*, 78(1):1–3, 1950.

A Philip Dawid. Present position and potential developments: Some personal views statistical theory the prequential approach. *Journal of the Royal Statistical Society: Series A (General)*, 147(2): 278–290, 1984.

Cynthia Dwork, Michael P Kim, Omer Reingold, Guy N Rothblum, and Gal Yona. Learning from outcomes: Evidence-based rankings. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 106–125. IEEE, 2019.

Cynthia Dwork, Michael P Kim, Omer Reingold, Guy N Rothblum, and Gal Yona. Outcome indistinguishability. *STOC*, 2021.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, pp. 1321–1330. PMLR, 2017.

Ursula Hébert-Johnson, Michael Kim, Omer Reingold, and Guy Rothblum. Multicalibration: Calibration for the (computationally-identifiable) masses. In *International Conference on Machine Learning*, pp. 1939–1948. PMLR, 2018.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

Christopher Jung, Changhwa Lee, Mallesh M Pai, Aaron Roth, and Rakesh Vohra. Moment multicalibration for uncertainty estimation. *arXiv preprint arXiv:2008.08037*, 2020.

Michael P Kim, Amirata Ghorbani, and James Zou. Multiaccuracy: Black-box post-processing for fairness in classification. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, pp. 247–254, 2019.

Jon Kleinberg, Sendhil Mullainathan, and Manish Raghavan. Inherent trade-offs in the fair determination of risk scores. *arXiv preprint arXiv:1609.05807*, 2016.

Volodymyr Kuleshov, Nathan Fenner, and Stefano Ermon. Accurate uncertainties for deep learning using calibrated regression. In *International Conference on Machine Learning*, pp. 2796–2804. PMLR, 2018.

Meelis Kull and Peter Flach. Novel decompositions of proper scoring rules for classification: Score adjustment as precursor to calibration. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 68–85. Springer, 2015.

Meelis Kull, Miquel Perello-Nieto, Markus Kängsepp, Hao Song, Peter Flach, et al. Beyond temperature scaling: Obtaining well-calibrated multiclass probabilities with dirichlet calibration. *arXiv preprint arXiv:1910.12656*, 2019.

Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2018.

John Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.

Geoff Pleiss, Manish Raghavan, Felix Wu, Jon Kleinberg, and Kilian Q Weinberger. On fairness and calibration. In *Advances in Neural Information Processing Systems*, pp. 5680–5689, 2017.

Eliran Shabat, Lee Cohen, and Yishay Mansour. Sample complexity of uniform convergence for multicalibration. *NeurIPS*, 2020.

Hao Song, Tom Diethe, Meelis Kull, and Peter Flach. Distribution calibration for regression. In *International Conference on Machine Learning*, pp. 5897–5906. PMLR, 2019.

Philipp Tschandl, Cliff Rosendahl, and Harald Kittler. The ham10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions. *Scientific data*, 5(1):1–9, 2018.

David Widmann, Fredrik Lindsten, and Dave Zachariah. Calibration tests in multi-class classification: A unifying framework. *arXiv preprint arXiv:1910.11385*, 2019.

Shengjia Zhao and Stefano Ermon. Right decisions from wrong predictions: A mechanism design alternative to individual calibration. In *International Conference on Artificial Intelligence and Statistics*, pp. 2683–2691. PMLR, 2021.

Shengjia Zhao, Tengyu Ma, and Stefano Ermon. Individual calibration with randomized forecasting. *arXiv preprint arXiv:2006.10288*, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Appendix.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Appendix.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See formal statements in Appendix D.
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix D.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Code is included in the supplemental material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] We stated most details; the rest can be found in the code.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] The figures have error bars.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] We use public datasets.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] We use public datasets.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]