Who Should Predict? Exact Algorithms For Learning to Defer to Humans

Hussein Mozannar
MIT-IBM Watson AI Lab
MIT, IDSS & CSAIL

Hunter Lang
MIT-IBM Watson AI Lab
MIT, CSAIL & IMES

Dennis Wei
MIT-IBM Watson AI Lab
IBM Research

Prasanna Sattigeri
MIT-IBM Watson AI Lab
IBM Research

Subhro Das
MIT-IBM Watson AI Lab
IBM Research

David Sontag
MIT-IBM Watson AI Lab
MIT, CSAIL & IMES

Abstract

Automated AI classifiers should be able to defer the prediction to a human decision maker to ensure more accurate predictions. In this work, we jointly train a classifier with a *rejector*, which decides on each data point whether the classifier or the human should predict. We show that prior approaches can fail to find a human-AI system with low misclassification error even when there exists a linear classifier and rejector that have zero error (the *realizable* setting). We prove that obtaining a linear pair with low error is NP-hard even when the problem is realizable. To complement this negative result, we give a mixed-integer-linear-programming (MILP) formulation that can optimally solve the problem in the linear setting. However, the MILP only scales to moderately-sized problems. Therefore, we provide a novel surrogate loss function that is realizable-consistent and performs well empirically. We test our approaches on a comprehensive set of datasets and compare to a wide range of baselines.

1 Introduction

AI systems are frequently used in combination with human decision-makers, including in high-stakes settings like healthcare (Beede et al., 2020). In these scenarios, machine learning predictors should be able to defer to a human expert instead of predicting on difficult or unfamiliar examples. However, when AI systems are used to provide a *second opinion* to the human, prior work shows that humans over-rely on the AI when it is incorrect (Jacobs et al., 2021; Mozannar et al., 2022), and these systems rarely achieve performance higher than either the human or AI alone (Liu et al., 2021a, Proposition 1). This motivates deferral-style systems, where *either* the classifier or the human predicts, to avoid over-reliance.

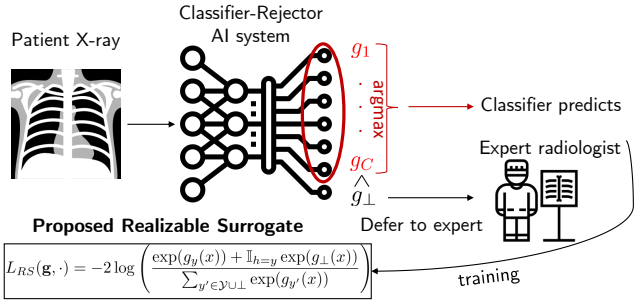


Figure 1: The learning to defer setting with the `RealizableSurrogate` illustrated in the application of making predictions for chest X-rays.

As a motivating example, suppose we want to build an AI system to predict the presence of pneumonia from a patient's chest X-ray, jointly with an human radiologist. The goal in this work is to jointly learn a classifier that can predict pneumonia and a *rejector*, which decides on each data point whether the classifier or the human should predict. By learning the classifier jointly with the rejector, the aim is for the classifier to *complement* the radiologist so that the Human-AI team performance is higher. We refer to the error rate of the Human-AI team as the *system error*.

Failure of Prior Approaches. Existing literature has focused on *surrogate loss functions* for deferral (Madras et al., 2018; Mozannar and Sontag, 2020; Verma and Nalisnick, 2022) and confidence based approaches (Raghu et al., 2019; Okati et al., 2021). We give a simple synthetic setting where

Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS) 2023, Valencia, Spain. PMLR: Volume 206.

all of these approaches fail to find a classifier/rejector pair with low system error. In this setting, there exists a halfspace classifier and halfspace rejector that have zero system error (illustrated in Figure 2), but our experiments in Section 7.2 demonstrate that all prior approaches fail to find a good classifier/rejector pair in this setting.

To understand possible reasons for this failure, we first study the **computational complexity** of learning with deferral using halfspaces for the rejector and the classifier, which we call LWD-H. The computational complexity of learning with deferral has received little attention in the literature. We prove that even in our simple setting where the data is realizable (i.e., there exists a halfspace classifier and halfspace rejector achieving zero system error), there is no polynomial-time algorithm that finds an approximately optimal pair of halfspaces unless $NP = RP$. We also extend our hardness result to approximation algorithms and when the data is not realizable by halfspaces. In contrast, training a linear classifier in the realizable linear setting can be solved in polynomial time with linear programming (Boyd and Vandenberghe, 2004).

Learning with deferral using halfspaces is also of significant *practical* importance. Sample efficiency is critical in learning with deferral since the training data is expensive to collect—it requires both human outputs *and* ground-truth labels. This motivates restricting to smaller model classes, and in particular to linear classifiers and rejectors. Linear models have the benefit of being interpretable with respect to the underlying features, which can be crucial for a human-AI deferral system. Additionally, the *head tuning* or *linear probing* paradigm, where only the final (linear) layer of a pretrained deep neural network is fine-tuned on different tasks, has become increasingly common as pretrained representations improve in quality, and it can be more robust than full fine-tuning (Kumar et al., 2022). However, as previously mentioned, existing surrogate approaches fail to find a good linear classifier and rejector even when one is guaranteed to exist. This motivates the need for an algorithm for **exact minimization** of the system training error.

We show that exact minimization of the system error can be formulated as a **mixed integer linear program** (MILP). This derivation overcomes a naive quadratic formulation of the problem. In addition to exactly minimizing the training loss, the MILP formulation allows us to easily incorporate constraints to govern the human-AI system. We show that modern commercial solvers such as Gurobi (Gurobi Optimization, LLC, 2022) are capable of solving fairly large instances of this MILP, making it a practical algorithm for the LWD-H problem. To obtain similar gains over prior approaches, but with a more scalable algorithm, we develop a new differentiable surrogate loss function L_{RS}, dubbed **`RealizableSurrogate`** , that can solve the LWD-H problem in the realizable setting by virtue of being *realizable-consistent* (Long and Servedio, 2013) for a large class of hypothesis sets that includes halfspace classifier/rejector pairs. We also show empirically that L_{RS} is competitive with prior work in the non-linear setting.

In section 3, we formalize the learning with deferral problem. We then study the computational complexity of LWD-H in section 4. We introduce our MILP approach in section 5 and our new surrogate **`RealizableSurrogate`** in section 6. In section 7, we evaluate our algorithms and baselines on a wide range of benchmarks in different domains, providing the most expansive evaluation of expert deferral algorithms to date. To summarize, the contributions of this paper are the following:

- **Computational Complexity of Deferral:** We prove the computational hardness of PAC-learning with deferral in the linear setting.
- **Mixed Integer Linear Program Formulation and `RealizableSurrogate` :** We show how to formulate learning to defer with halfspaces as a MILP and provide a novel surrogate loss.
- **Experimental Evaluation:** We showcase the performance of our algorithms on a wide array of datasets and compare them to several existing baselines. We contribute a publicly available repository with implementations of existing baselines and datasets: https://github.com/clinicalml/human_ai_deferral

2 Related Work

A natural baseline for the learning to defer problem is to first learn a classifier that minimizes average misclassification error, then learn a model that predicts the probability that the human makes an error on a given example, and finally defer if the probability that the classifier makes an error is higher than that of the human. This is the approach adapted by Raghu et al. (2019). However, this does not allow the classifier to adapt to the human. Another natural approach is to model this problem as a mixture of experts: the human and the AI. This is the approach introduced by Madras et al. (2018) and adapted by Wilder et al. (2020); Pradier et al. (2021) by introducing mixture of experts surrogates. However, this approach has been to shown to fail empirically as the loss is not easily amenable to optimization. Subsequent work (Mozannar and Sontag, 2020) introduced consistent surrogate loss functions for the learning with deferral objective, with follow-up approaches addressing limitations including better calibration (Raman and Yee, 2021; Liu et al., 2021b). Charusaie et al. (2022) provides a family of convex surrogates for learning with deferral that generalize prior approaches, however, our proposed surrogate does not belong to that family. Another consistent convex surrogate was proposed by Verma and Nalisnick (2022) via a one-vs-all approach, it can be shown

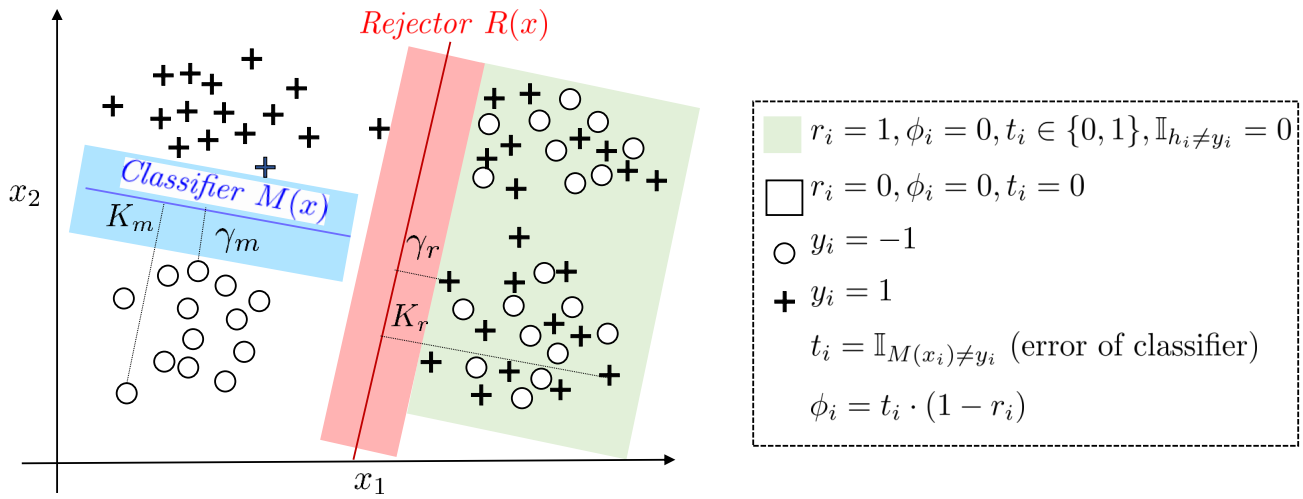


Figure 2: The realizable LWD-H setting illustrated. The task is binary classification with labels $\{o, +\}$; the human is perfect on the green-shaded region, and the data outside the green region is linearly separable. As a result, the optimal classifier and rejector obtain zero error. Assumption 2 is illustrated graphically as well as the MILP variables of equations (8)-(13).

that it belongs to the family proposed by (Charusaie et al., 2022). Keswani et al. (2021) proposes a surrogate loss which is the sum of the loss of learning the classifier and rejector separately but that is not a consistent surrogate. De et al. (2020) proved hardness of linear regression (not classification) where some training points are allocated to the human (not deferral but subset selection of the training data). Finally, Okati et al. (2021) proposes a method that iteratively optimizes the classifier on points where it outperforms the human on the training sample, and then learns a post-hoc rejector to predict who between the human and the AI has higher error on each point. The setting when the cost of deferral is constant, has a long history in machine learning and goes by the name of rejection learning (Cortes et al., 2016; Chow, 1970; Bartlett and Wegkamp, 2008; Charoenphakdee et al., 2021) or selective classification (only predict on $x\%$ of data) (El-Yaniv and Wiener, 2010; Geifman and El-Yaniv, 2017; Gangrade et al., 2021; Acar et al., 2020). Our MILP formulation is inspired by work in binary linear classification that optimizes the 0-1 loss exactly (Ustun and Rudin, 2016; Nguyen and Sanner, 2013).

3 Learning with Deferral: Problem Setup

We frame the learning with deferral setting as the task of predicting a target $Y \in \mathcal{Y} = \{1, \cdots, C\}$. The classifier has access to features $X \in \mathcal{X} = \mathbb{R}^d$, while the human (also referred to as the expert) has access to a potentially different set of features $Z \in \mathcal{Z}$ which may include side-information beyond X. The human is modeled as a fixed predictor $h : \mathcal{Z} \to \mathcal{Y}$. The AI system consists of a classifier $m : \mathcal{X} \to \mathcal{Y}$ and a rejector $r : \mathcal{X} \to \{0, 1\}$. When $r(x) = 1$, the prediction is deferred to the human and we incur a cost if the human makes an error, plus an additional, optional penalty term: $\ell_{\text{HUM}}(x, y, h) = \mathbb{I}_{h \neq y} + c_{\text{HUM}}(x, y, h)$. When $r(x) = 0$, then the classifier makes the final decision and incurs a cost with a different optional penalty term: $\ell_{\text{AI}}(x, y, m) = \mathbb{I}_{m \neq y} + c_{\text{AI}}(x, y, m)$. We can put this together into a loss function for the classifier and rejector:

$$L_{\text{def}}(m, r) = \mathbb{E}_{X,Y,Z}\left[\ell_{\text{AI}}\big(X, Y, m(X)\big) \cdot \mathbb{I}_{r(X)=0} + \ell_{\text{HUM}}(X, Y, h(Z)) \cdot \mathbb{I}_{r(X)=1}\right]. \tag{1}$$

In this paper we focus mostly on the cost of misclassification with no additional penalties, the deferral loss becomes a misclassification loss $L^{0-1}_{\text{def}}(m, r)$ for the human-AI system, and the optimization problem is:

$$\underset{m,r}{\text{minimize}}\; L^{0-1}_{\text{def}}(m, r) := \mathbb{P}\left[((1 - r(X))\, m(X) + r(X) h(Z)) \neq Y\right]. \tag{2}$$

Data. We assume access to samples $S = \{(x_i, h(z_i), y_i)\}_{i=1}^n$ where $h(z_i)$ is the human's prediction on the example, but note that we do not observe z_i, the information used by the human. We emphasize that the label y_i and human prediction $h(z_i)$ are different, even though y_i could also come from humans. For example in our chest X-ray classification example, y_i could come from a consensus of 3 or more radiologists, while $h(z_i)$ is the prediction of a single radiologist not involved with the label. Given the dataset S the system *training loss* is given by:

$$\hat{L}^{0-1}_{\text{def}}(m, r) := \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}_{m(x_i) \neq y_i} \mathbb{I}_{r(x_i)=0} + \mathbb{I}_{h(z_i) \neq y_i} \mathbb{I}_{r(x_i)=1} \tag{3}$$

In the following section, we study the computational complexity of learning with deferral using halfspaces, which reduces to studying the optimization problem (3) when m and r are constrained to be halfspaces.

4 Computational Complexity of Learning with Deferral

The misclassification error of the human-AI team in equation (2) is challenging to optimize as it requires searching over a joint set of functions for the classifier and rejector, in addition to dealing with the nonconvex 0-1 aspect. To study the computational complexity of minimizing the loss, we restrict our attention to a foundational setting: linear classifiers and linear rejectors in the binary label scenario.

We begin with the realizable case when there exists a halfspace classifier and rejector that can achieve zero loss:

Assumption 1 (Realizable Linear Setting)**.** Let $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \{0, 1\}$. We assume that for the given expert h there exists a linear classifier $m^*(x) = \mathbb{I}_{M^\top x > 0}$ and a linear rejector $r^*(x) = \mathbb{I}_{R^\top x > 0}$ that achieve 0 error:

$$\mathbb{E}_{(x,y,z)\sim\mathbf{P}} \left[\mathbb{I}_{m^*(x)\neq y}\mathbb{I}_{r^*(x)=0} + \mathbb{I}_{h(z)\neq y}\mathbb{I}_{r^*(x)=1} \right] = 0.$$

This setting is illustrated in Figure 2. Since the decision regions of m and r are halfspaces, we also use the term "halfspace" interchangeably. Note that while the classifier is assumed to be linear, the human can have a non-linear decision boundary. The analog of this assumption in the binary classification without deferral setting is to assume that there exists a halfspace that can correctly classify all the data points. In that case, we can formulate the optimization problem as a linear program to efficiently find the optimal solution (Boyd and Vandenberghe, 2004).

Hardness. In contrast to learning without deferral, we will prove that in general, it is computationally hard to learn a linear m and r under Assumption 1. Define the *learning with deferral using halfspaces* (LWD-H) problem as that of finding halfspace m and halfspace r such that the system error in (2) is approximately minimized.

Theorem 1. *Let $\epsilon > 0$ be an arbitrarily small constant. Under a guarantee that there exist halfspaces m^*, r^* with zero system loss (Assumption 1), there is no polynomial-time algorithm to find a pair of classifier-rejector halfspaces with error $1/2 - \epsilon$ unless $NP = RP$.*

This shows that even in the realizable setting (i.e., there exists a pair of halfspaces with zero system loss), it is hard to find a pair of halfspaces that even gets system error $1/2 - \epsilon$.

Corollary 1. *There is no efficient proper PAC-learner for LWD-H unless $NP = RP$.*

Proof Sketch. First, because the true distribution of points could be supported on a finite set, the LWD-H problem boils down to approximately minimizing the training loss (3). Our proof relies on a reduction from the problem of learning an intersection of two halfspaces in the realizable setting. Let $D = \{x_i, y_i\}_{i=1}^n$ and suppose there exists an intersection of two half-spaces g_1, g_2 that achieve 0 error for D. This is an instance of learning an intersection of two halfspaces in the realizable setting, which is hard to even *weakly* learn (Khot and Saket, 2011). We show that this is an instance of the realizable LWD-H problem by setting $m = g_1$ and $r = \bar{g_2}$ and the human H to always predict 0. Hence, an algorithm for efficiently finding a classifier/rejector pair with error $\frac{1}{2} - \epsilon$ would also find an intersection of halfspaces with error $\frac{1}{2} - \epsilon$, which is hard unless $NP = RP$. □

All proofs can be found in the Appendix. This hardness result holds in the realizable setting, with proper learning, and with no further assumptions on the data distribution.

Extensions. Even if the problem is not realizable and the goal is to find an approximation algorithm, this is still computationally hard as presented in the following corollary.

Corollary 2. *When the data is not realizable (i.e., Assumption 1 is violated), there is no polytime algorithm for finding a pair of halfspaces with error $\frac{1}{2} - \epsilon$ unless $NP = RP$.*

Exact Solution. These hardness results motivate the need for new approaches to solving the LWD-H problem. In the next section, we derive a scheme to exactly minimize the misclassification error of the human/AI system using mixed-integer linear programming (MILP).

5 Mixed Integer Linear Program Formulation

In the previous section, we saw that in the linear setting it is computationally hard to learn an optimal classifier and rejector pair. As discussed in Section 1, we are interested in the linear setting due to the cost of labeling large datasets for learning with deferral. Linear predictors can perform similar to non-linear predictors in applications involving high-dimensional medical data (Razavian et al., 2015). Moreover, we can rely on pre-trained representations, which can allow linear predictors on top of embedded representations to attain performance comparable to non-linear predictors (Bengio et al., 2013).

A First Formulation. As a first step, we write down a mixed integer *nonlinear* program for the optimization of the training loss $\hat{L}_{\text{def}}^{0-1}$ in (3) over linear classifiers and linear rejectors with binary labels. For simplicity, let $\mathcal{Y} = \{-1, +1\}$. A direct translation of (3) with halfspace classifiers and rejectors yields the following:

$$M^*, R^*, \cdot = \underset{M,R,m_i,r_i}{\arg\min} \sum_{i=1}^{n} (1 - r_i)\mathbb{I}_{m_i \neq y_i} + r_i \mathbb{I}_{h_i \neq y_i} \quad (4)$$

$$\text{s.t.} \quad m_i = \text{sign}(M^\top x_i), \;\; r_i = \mathbb{I}_{R^\top x_i \geq 0} \; \forall i \in [n], \quad (5)$$
$$M \in \mathbb{R}^d, R \in \mathbb{R}^d.$$

The variables m_i and r_i are simply the binary outputs of the classifier and rejector. We observe that the objective involves

a quadratic interaction between the classifier and rejector. Furthermore, the constraints (5) are indicator constraints that are difficult to optimize.

Making Objective Linear. We observe that since the r_i's are binary, the term $(1-r_i)\mathbb{I}_{m_i \neq y_i}$ can be equivalently rewritten as $\max(0, \mathbb{I}_{m_i \neq y_i} - r_i)$. This is a crucial simplification that avoids having a mixed integer quadratic program. Below we use this to create a binary variable $t_i = \mathbb{I}_{m_i \neq y_i}$ representing the error of the classifier and a second *continuous* variable ϕ_i that upper bounds $\max(0, t_i - r_i)$ and represents the classifier error after accounting for deferral.

Making Constraints Linear. Constraints (5) make sure that the binary variables r_i and m_i are the predictions of half-spaces R and M respectively. As mentioned above, we will formulate the problem using the classifier error variables t_i instead of the classifier predictions m_i. To reformulate constraints (5) in a linear fashion, we have to make assumptions on the optimal M and R:

Assumption 2 (Margin)**.** The optimal solution (M, R) that minimizes the training loss (3) has margin and is bounded, meaning that (M, R) satisfy the following for all $i \in [n]$ in the training set and some $\gamma_m, \gamma_r, K_m, K_r > 0$:

$$\gamma_m \leq |M^\top x_i| \leq K_m - \gamma_m, \ \gamma_r \leq |R^\top x_i| \leq K_r - \gamma_r \quad (6)$$

A similar assumption is made in (Ustun and Rudin, 2016). The upper bounds in (6) are often naturally satisfied as we usually deal with bounded feature sets $\mathcal{X}$ such that we can normalize x_i to have unit norm, and the norms of M and R are constrained for regularization.

Mixed Integer Linear Program. With the above ingredients and taking inspiration from the big-M approach of Ustun and Rudin (2016), we can write down the resulting mixed integer linear program (MILP):

$$M^*, R^*, \cdots = \underset{M,R,\{r_i\},\{t_i\},\{\phi_i\}}{\arg\min} \sum_i \phi_i + r_i \mathbb{I}_{h_i \neq y_i}, \ s.t. \quad (7)$$

$$\phi_i \geq t_i - r_i, \qquad \phi_i \geq 0 \quad \forall i \in [n] \quad (8)$$

$$K_m t_i \geq \gamma_m - y_i M^\top x_i \quad \forall i \in [n] \quad (9)$$

$$R^\top x_i \leq K_r r_i + \gamma_r (r_i - 1), \quad (10)$$

$$R^\top x_i \geq K_r (r_i - 1) + \gamma_r r_i \quad \forall i \in [n] \quad (11)$$

$$r_i \in \{0,1\}, t_i \in \{0,1\}, \quad (12)$$

$$\phi_i \in \mathbb{R}^+ \quad \forall i \in [n], \ M, R \in \mathbb{R}^d \quad (13)$$

Please see Figure 2 for a graphical illustration of the variables. We show that constraints (11) function as intended; the rest of the constraints are verified in the Appendix. When $r_i = 0$, then we have the constraints $R^\top x_i \leq -\gamma_r$ and $R^\top x_i \geq -K_r$: this correctly forces the rejector to be negative. When $r_i = 1$, we have $R^\top x_i \geq \gamma_r$ and $R^\top x_i \leq K_r$: which means the rejector is positive. Note that we do not need to know the margin γ_r exactly, only a lower bound γ, $0 < \gamma \leq \gamma_r$; the formulation is still correct with γ in place of γ_r. However, we cannot set $\gamma = 0$ as then the trivial solution $R = 0$ is feasible and the constraint is void. The same statements apply to γ_m. This MILP has $2n$ binary variables, $n + 2d$ continuous variables and $4n$ constraints. Finally, note that the MILP minimizes the 0-1 error even when Assumption 1 is violated.

Regularization and Extension to Multiclass. We can add l_1 regularization to our model by adding the l_1 norm of both M and R to the objective. This is done by defining two sets of variables constrained to be the l_1 norm of the classifier and rejector and adding their values to the objective in (8). Adding regularization can help prevent the MILP solution from overfitting to the training data. The above MILP only applies to binary labels, but can be generalized to the multiclass setting where $\mathcal{Y} = \{1, \cdots, C\}$ (see Appendix).

Generalization Bound. Under Assumption 2 and non-realizability, assume $\|x_i\|_1 \leq 1$ and constrain the search of the MILP to M and R with infinity norms of at most K_m and K_r respectively. We can relate the performance of MILP solution on the training set to the population 0-1 error.

Proposition 1. *For any expert h and data distribution $\mathbf{P}$ over $\mathcal{X} \times \mathcal{Y}$ that satisfies Assumption 2, let $0 < \delta < \frac{1}{2}$. Then with probability at least $1 - \delta$, the following holds for the empirical minimizers $(\hat{m}^*, \hat{r}^*)$ obtained by the MILP:*

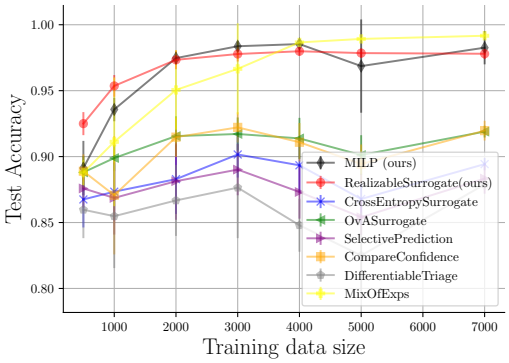
$$L_{0-1}(\hat{m}^*, \hat{r}^*) \leq \hat{L}^{0-1}_{\text{def}}(\hat{m}^*, \hat{r}^*) + \frac{(K_m + K_r) d \sqrt{2 \log d} + 10\sqrt{\log(2/\delta)}}{\sqrt{n \mathbb{P}(h(Z) \neq Y)}}.$$

This bound improves on surrogate optimization since the MILP will achieve a lower training error, $\hat{L}^{0-1}_{\text{def}}(\hat{m}^*, \hat{r}^*)$, than the surrogate, which optimizes a different objective.

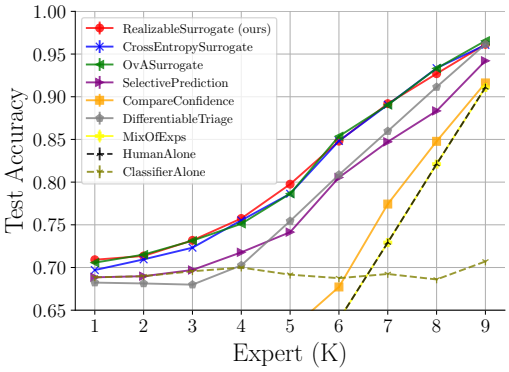
Adding Constraints. A major advantage of the MILP formulation is that it allows us to provably integrate any linear constraints on the variables with ease. For example, the constraints mentioned in Section 3 can be added to the MILP as follows in a single constraint:

- Coverage: $\sum_i r_i / n \leq \beta$
- Fairness: $\sum_{i:A=1} (\phi_i + r_i \mathbb{I}_{h_i \neq y_i}) / |\{i : A = 1\}| = \sum_{i:A=0} (\phi_i + r_i \mathbb{I}_{h_i \neq y_i}) / |\{i : A = 0\}|$.

So far, we have provided an exact solution to the linear learning to defer problem. However, the MILP requires significant computational time to find an exact solution for large datasets. Moreover, we might need a non-linear classifier or rejector to achieve good error. The remaining questions are (i) how to efficiently find a good pair of halfspaces



(a) Synthetic Data Sample Complexity



(b) CIFAR-K Semi-Synthetic

Figure 3: (a) Test performance of the different methods on synthetic data as we increase the training data size and repeat the randomization over 10 trials to get standard errors. (b) Test performance on the semi-synthetic CIFAR-K dataset vs. the number of classes K for which the expert is perfect.

for large datasets and (ii) how to generalize to non-linear predictors. In the following section, we give a novel surrogate loss function that is optimal in the realizable LWD-H setting, performs well with non-linear predictors, and can be efficiently minimized (to a local optimum).

6 Realizable Consistent Surrogate

Most of machine learning practice is based on optimizing surrogate loss functions of the true loss that one cares about. The surrogate loss functions are chosen so that optimizing them also optimizes the true loss functions, and also chosen to be differentiable so that they are readily optimized. This first property is captured by the notion of *consistency*, which was the main focus of much of the prior work on expert deferral: (Mozannar and Sontag, 2020; Verma and Nalisnick, 2022; Charusaie et al., 2022). We start by giving a formal definition of the consistency of a surrogate loss function:

Definition 1 (Consistency[1])**.** *A surrogate loss function $\widetilde{L}(m,r)$ is a consistent loss function for another loss $L_{\text{def}}^{0-1}(m,r)$ if optimizing the surrogate over all measurable functions is equivalent to minimizing the original loss.*

For example, the surrogates L_{CE} and Ψ_{OvA} both satisfy consistency for $L_{\text{def}}^{0-1}(m,r)$ (Mozannar and Sontag, 2020; Verma and Nalisnick, 2022). It is crucial to note that consistency only applies when optimizing over *all measurable functions*. Conversely, in LWD-H, and in the setting of Figure 2, when we optimize with linear functions, consistency does not provide any guarantees, which explains why these methods can fail in that setting.

Since we normally optimize over a restricted model class, we want our guarantee for the surrogate to also hold for optimization under a certain model class. The notion of realizable $\mathcal{H}$-consistency is such a guarantee that has proven fruitful for classification (Long and Servedio, 2013; Zhang and Agarwal, 2020) and was extended by Mozannar and Sontag (2020) for learning with deferral. We recall the notion when extended for learning with deferral:

Definition 2 (realizable $(\mathcal{M},\mathcal{R})$-consistency)**.** *A surrogate loss function $\widetilde{L}(m,r)$ is a realizable $(\mathcal{M},\mathcal{R})$-consistent loss function for the loss $L_{\text{def}}^{0-1}(m,r)$ if there exists a zero error solution $m^*, r^* \in \mathcal{M}\times\mathcal{R}$ with $L_{\text{def}}^{0-1}(m^*,r^*)=0$. Then optimizing the surrogate returns such zero error solution:*

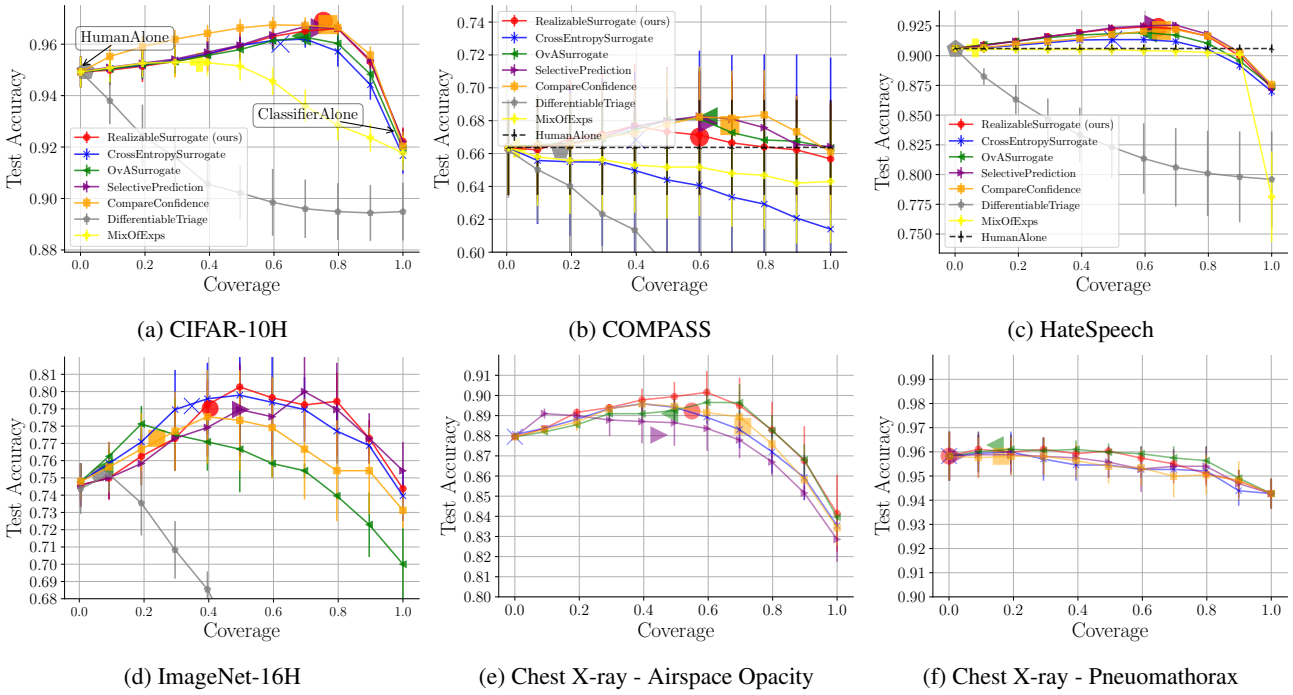
$$\widetilde{m},\widetilde{r} \in \operatorname*{arg\,inf}_{m,r\in\mathcal{M}\times\mathcal{R}} \widetilde{L}(m,r) \implies L_{\text{def}}^{0-1}(\widetilde{m},\widetilde{r})=0$$

Realizable $(\mathcal{M},\mathcal{R})$-consistency guarantees that when our data comes from some ground-truth $m^*, r^* \in \mathcal{M}\times\mathcal{R}$, then minimizing the (population) surrogate loss will find an optimal (m,r) pair. We propose a novel, differentiable, and $(\mathcal{M},\mathcal{R})$-consistent surrogate for learning with deferral when $\mathcal{M}$ and $\mathcal{R}$ are *closed under scaling*. A class $\mathcal{G}$ of scoring functions from $\mathcal{X}$ to $\mathbb{R}^C$ is closed under scaling if $\boldsymbol{g}\in\mathcal{G} \implies \alpha\boldsymbol{g}\in\mathcal{G}$ for any $\alpha\in\mathbb{R}$. For example, we can let $\mathcal{G}$ be the class of linear scoring functions $\boldsymbol{g}(x)=G^\top x$ and $G\in\mathbb{R}^{d\times C}$. Our results hold for arbitrary $\mathcal{G}$ that are closed under scaling, e.g., ReLU feedforward neural networks (FNN). We parameterize the (m,r) pair with $|\mathcal{Y}|+1$ dimensional scoring function $\boldsymbol{g}:(g_1,\ldots,g_{|\mathcal{Y}|},g_\perp)$. We define $m(x)=\arg\max_{y\in\mathcal{Y}} g_y(x)$ and $r(x)=\mathbb{I}_{\max_{y\in\mathcal{Y}} g_y(x)\le g_\perp(x)}$. The joint classifier-rejector model class $(\mathcal{M},\mathcal{R})$ is thus defined by $\mathcal{G}$, and we say $(\mathcal{M},\mathcal{R})$ is closed under scaling whenever $\mathcal{G}$ is closed under scaling. The proposed new surrogate loss L_{RS}, dubbed `RealizableSurrogate`, is defined at each point (x,y,h) as:

$$L_{RS}(\mathbf{g},\cdot) = -2\log\left(\frac{\exp(g_y(x))+\mathbb{I}_{h=y}\exp(g_\perp(x))}{\sum_{y'\in\mathcal{Y}\cup\perp}\exp(g_{y'}(x))}\right)$$

The surrogate is illustrated in Figure 1. Notice that when the human is incorrect, i.e. $\mathbb{I}_{h=y}=0$, the loss incentivizes

[1]This is also referred to as Fisher consistency (Lin, 2002) and classification-calibration (Bartlett et al., 2006).



(a) CIFAR-10H (b) COMPASS (c) HateSpeech

(d) ImageNet-16H (e) Chest X-ray - Airspace Opacity (f) Chest X-ray - Pneuomathorax

Figure 4: Accuracy vs coverage (fraction of points where classifier predicts) plots across the real world datasets showcasing the behavior of our method and the baselines. On each plot, we showcase the test accuracy of each method with a large marker, with the curve representing varying the rejector threshold on the test set. To achieve different levels of coverage, we sort the rejection score for each method on the test set and vary the threshold used, for `RealizableSurrogate` the rejector is defined as $r(x) = \mathbb{I}_{g_\perp(x) - \max_y g_y(x) \geq c}$ where the optimal solution is at $c = 0$ and we vary $c \in \mathbb{R}$ to obtain the curve.

the classifier to be correct, similar to cross entropy loss. However, when the human is correct, the learner has the *choice* to either fit the target or defer: there is no penalty for choosing to do one or the other. This is what enables the classifier to complement the human and differentiates L_{RS} from prior surrogates, such as L_{CE} (Mozannar and Sontag, 2020), that are not realizable-consistent (see Appendix) and penalize the learner for not fitting the target even when deferring. This property is showcased by the fact that our surrogate is realizable $(\mathcal{M}, \mathcal{R})$-consistent for model classes that are closed under scaling. Moreover, it is an upper bound of the true loss $L^{0-1}_{\text{def}}(m, r)$. The theorem below characterizes the properties of our novel surrogate function.

Theorem 2. *The* `RealizableSurrogate` L_{RS} *is a realizable* $(\mathcal{M}, \mathcal{R})$*-consistent surrogate for* L^{0-1}_{def} *for model classes closed under scaling, and satisfies* $L^{0-1}_{\text{def}}(m, r) \leq L_{RS}(m, r)$ *for all* (m, r).

This theorem implies that when Assumption 1 is satisfied and $\mathcal{G}$ is the class of linear scoring functions, minimizing L_{RS} yields a classifier-rejector pair with zero system error. The resulting classifier is the halfspace $\mathbb{I}((G_1 - G_0)^\top x \geq 0)$ and the form of the rejector is $\mathbb{I}((G_\perp^\top x - \max(G_1^\top x, G_0^\top x)) \geq 0)$, which is an intersection of halfspaces. However, one can see that by setting $G_0 = 0$ and optimizing over only G_1 and $G_\perp$ we can recover a linear classifier and linear rejector; in practice we only do this when we explicitly want a linear rejector.

The surrogate is differentiable but *non*-convex in $\mathbf{g}$, though it is convex in each g_i. Indeed, a jointly convex surrogate that provably works in the realizable linear setting would contradict Theorem 1. In practice, we observe that in the linear realizable setting, the local minima reached by gradient descent obtain zero training error despite the nonconvexity. The mixture-of-experts surrogate in Madras et al. (2018) is realizable $(\mathcal{M}, \mathcal{R})$-consistent, non-convex and not classification consistent as shown by Mozannar and Sontag (2020), however, Mozannar and Sontag (2020) also showed that it leads to worse empirical results than simple baselines. We have not been able to prove or disprove that `RealizableSurrogate`is classification-consistent, unlike other surrogates like that of Mozannar and Sontag (2020). It remains an open problem to find both a consistent **and** a realizable-consistent surrogate.

Underfitting the target. Minimizing the proposed loss leads to a classifier that attempts to complement the human. One consequence is that the classifier might have high error on points that are deferred to the human, resulting in possibly high error across a large subset of the data domain. We can explicitly encourage the classifier to fit the target on all

points by adding an extra term to the loss:

$$L^{\alpha}_{RS}(\mathbf{g}, x, y, h) = -\alpha \log\left(\frac{\exp(g_y(x)+\mathbb{I}_{h=y}\exp(g_\perp(x))}{\sum_{y'\in\mathcal{Y}\cup\perp}\exp(g_{y'}(x))}\right) - (1-\alpha)\log\left(\frac{\exp(g_y(x))}{\sum_{y'\in\mathcal{Y}}\exp(g_{y'}(x))}\right) \quad (14)$$

The new loss L^{α}_{RS} with $\alpha \in [0,1]$ (a hyperparameter) is a convex combination of L_{RS} and the cross entropy loss for the classifier (with the softmax applied only over the functions g_y rather than including $g_\perp$). Empirically, this allows the points that are deferred to the human to still help provide extra training signal to the classifier, which is useful for sample-efficiency when training complex, non-linear hypotheses. Finally, due to adding the parameter α, the loss no longer remains realizable consistent, thus we let the rejector be $r(x) = \mathbb{I}_{g_\perp(x)-\max_y g_y(x)\geq\tau}$ and we learn τ with a line search to maximize system accuracy on a validation set. In the next section, we evaluate our approaches with an extensive empirical benchmark.

7 Experiments

7.1 Human-AI Deferral Benchmark

Objective. We investigate the empirical performance of our proposed approaches compared to prior baselines on a range of datasets. Specifically, we want to compare the accuracy of the human-AI team at the learned classifier-rejector pairs. We also check the accuracy of the system when we change the deferral policy by varying the threshold used for the rejector, this leads to an accuracy-coverage plot where *coverage* is defined as the fraction of the test points where the classifier predicts.

Datasets. In Table 1 we list the datasets used in our benchmark. We start with synthetic data described below, then semi-synthetic data with CIFAR-K (Mozannar and Sontag, 2020). We then evaluate on 5 real world datasets with three image classification domains with multiple tasks per domain, a natural language domain and a tabular domain. Each dataset is randomly split 70-10-20 for training-validation-testing respectively.

Baselines. We compare to multiple methods from the literature including: the confidence method from Raghu et al. (2019) (CompareConfidence), the surrogate L^{α}_{CE} from Mozannar and Sontag (2020) (CrossEntropySurrogate), the surrogate Ψ_{OvA} from Verma and Nalisnick (2022) (OvASurrogate), Diff-Triage from Okati et al. (2021) (DifferentiableTriage), mixture of experts from Madras et al. (2018) (MixOfExps) and finally a selective prediction baseline that thresholds classifier confidence for the rejector (SelectivePrediction). For all baselines and datasets, we train using Adam and use the same learning rate and number of training epochs to ensure an equal footing across baselines, each run is repeated for 5 trials with different dataset splits. We track the best model in terms of system accuracy on a validation set for each training epoch and return the best performing model. For `RealizableSurrogate` , we perform a hyperparameter search on the validation set over $\alpha \in [0,1]$, and do hyperparameter tuning over L^{α}_{CE}.

7.2 Synthetic and Semi-Synthetic Data

Synthetic Data. We create a set of synthetic data distributions that are realizable by linear functions (or nearly so) to benchmark our approach. For the input X, we set the dimension d, and experiment with two data distributions. (1) Uniform distribution: we draw points $X \sim \text{Unif}(0,U)^d$ where $U \in \mathbb{R}^+$; (2) Mixture-of-Gaussians: we fix some $K \in \mathbb{N}$ and generate data from K equally weighted Gaussians, each with random uniform means and variances. To obtain labels Y that satisfy Assumption 1, we generate two random halfspaces and denote one as the optimal classifier $m^*(x)$ and the other as the optimal rejector $r^*(x)$. We then set the labels Y on the side where $r^*(x) = 0$ to be consistent with $m^*(x)$ with probability $1-p_m$ and otherwise uniform. When $r^*(x) = 1$, we sample the labels uniformly. Finally, we choose the human expert to have error p_{h0} when $r^*(x) = 0$ and have error p_{h1} when $r^*(x) = 1$. When $p_m = 0, p_{h0} \in [0,1]$, and $p_{h1} = 0$, this process generates datasets $D = \{x_i, y_i, h_i\}_{i=1}^n$ that satisfy Assumption 1.

Sample Complexity. For realizable data with a feature distribution that is mixture of Gaussians ($d = 30$, $p_m = 0, p_{h0} = 0.3, p_{h1} = 0$), Figure 3a plots the test accuracy of the different methods on a held-out dataset of 5k points as we increase the training data size. We observe that MILP and `RealizableSurrogate` are able to get close to zero error, while all other methods fail at finding a near zero-error solution. We also experiment with non-realizable data. For example, when $p_m = 0.1, p_{h0} = 0.4, p_{h1} = 0.1$ with $n = 1000$, the optimal test error is $7.5 \pm 1.0\%$ for the generated data: the MILP obtains 11.2 error and `RealizableSurrogate` achieves 17.8 ± 1.0 error, while the best baseline CrossEntropySurrogate achieves 21.4 ± 1.1 error. In the Appendix, we show results on the uniform data distribution, which shows an identical pattern, and we study the run-time and performance of the MILP as we increase the error probabilities.

CIFAR-K. We use the CIFAR-10 image classification dataset (Krizhevsky et al., 2009) and employ a simple convolution neural network (CNN) with three layers. We consider the human expert models from Mozannar and Sontag (2020); Verma and Nalisnick (2022): if the image is in the first K classes the expert is perfect, otherwise the expert predicts randomly. Figure 3b shows the test accuracy of the different methods as we vary the expert strength K. `RealizableSurrogate` outperforms the second best method by 0.8% on average and up to 2.8% maximum showcasing that the method can perform well for non-linear

Table 1: Datasets used for our benchmark for learning with deferral to humans. We note the total number of samples n, the target set size $|\mathcal{Y}|$, the number of tasks in each dataset (a task is a set of human and target labels), the human expert where 'random annotator' means that for each point we have multiple human annotations and we let the target be a consensus and the human label be a random sample while 'separate human annotation' means that the human label is completely separate from the label annotations and finally the model class for both the classifier and rejector.

Dataset	n	$\lvert\mathcal{Y}\rvert$	Number of Tasks	Human	Model Class
SyntheticData (ours)	arbitrary	2	1	synthetic	linear
CIFAR-K	60k	10	10 (per expert k)	synthetic (perfect on k classes)	CNN
CIFAR-10H (Battleday et al., 2020)	10k	10	1	separate human annotation	pretrained WideResNet (Zagoruyko and Komodakis, 2016)
Imagenet-16H (Kerrigan et al., 2021)	1.2k	16	4 (per noise version)	separate human annotation	pretrained DenseNet121 (Huang et al., 2017), fine-tuning last layer only
HateSpeech (Davidson et al., 2017)	25k	3	1	random annotator	FNN on embeddings from SBERT (Reimers and Gurevych, 2019)
COMPASS (Dressel and Farid, 2018)	1k	2	1	separate human annotation	linear
NIH Chest X-ray (Wang et al., 2017; Majkowska et al., 2020)	4k	2	4 (for different conditions)	random annotator	pretrained DenseNet121 on non-human labeled data

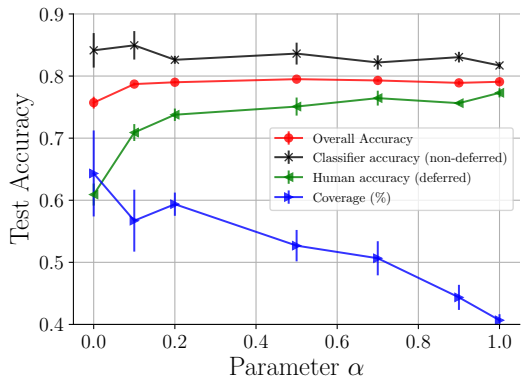


Figure 5: Different metrics: overall accuracy, accuracy when we defer, accuracy when we don't and coverage, as we vary the hyper parameter α in the `RealizableSurrogate` surrogate loss.

predictors.

Hyperparameter α. We show how the behavior of the classifier and rejector system changes when we modify the hyperparameter $\alpha \in [0, 1]$ in Figure 5. When α is small, the behavior of the surrogate is the same as selective prediction which is why we see the lowest accuracy of the human when we defer. As α increases to 1, we can see that the system better adapts to the human.

7.3 Realistic Data

Models. In Figure 4, we showcase the test accuracy of the different baselines on the real datasets in Table 1, and illustrate their behavior when we constrain our method and the baselines to achieve different levels of coverage. We can see that L_{RS}^{α} is competitive with the best baseline on each dataset/task. Moreover, we see that the human-AI team is often able to achieve performance that is higher than the human or classifier on their own. The methods often achieve peak performance at a coverage rate that is not at the extremes of [0,1], and on each of the six datasets we notice variability between the peak accuracy coverage rate indicating tat they are finding different solutions. This demonstrates that deferral using L_{RS}^{α} is able to achieve complementary human-AI team performance in practice. In summary, the new surrogate L_{RS} performs as well as the MILP on synthetic data, and as well as all the baselines (or better) on real-world data.

8 Discussion

We have shown that properly learning halfspaces with deferral (LWD-H) is computationally hard and that existing approaches in the literature fail in this setting. Understanding the computational limits of learning to defer led to the design of a new exact algorithm (the MILP) and a new surrogate (`RealizableSurrogate`) that both obtain better empirical performance than existing surrogate approaches. Studying $(\mathcal{M}, \mathcal{R})$-consistency in the non-realizable setting, obtaining conditions under which nonconvex surrogates like L_{RS} can be provably and efficiently minimized, and considering *online* versions of learning to defer are interesting directions for future work. As human-AI teams are deployed in real world decision-making scenarios, better and safer methods for training these systems are of critical interest. Giving the AI the power to allow the human to predict or not requires very careful optimization of the rejector so that we have favorable outcomes, this motivates the need for exact algorithms with guarantees.

Acknowledgments

HM is thankful for the support of the MIT-IBM Watson AI Lab.

References

Acar, D. A. E., Gangrade, A., and Saligrama, V. (2020). Budget learning via bracketing. In *International Conference on Artificial Intelligence and Statistics*, pages 4109–4119. PMLR.

Bartlett, P. L., Jordan, M. I., and McAuliffe, J. D. (2006). Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101(473):138–156.

Bartlett, P. L. and Wegkamp, M. H. (2008). Classification with a reject option using a hinge loss. *Journal of Machine Learning Research*, 9(Aug):1823–1840.

Battleday, R. M., Peterson, J. C., and Griffiths, T. L. (2020). Capturing human categorization of natural images by combining deep networks and cognitive models. *Nature communications*, 11(1):1–14.

Beede, E., Baylor, E., Hersch, F., Iurchenko, A., Wilcox, L., Ruamviboonsuk, P., and Vardoulakis, L. M. (2020). A human-centered evaluation of a deep learning system deployed in clinics for the detection of diabetic retinopathy. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–12.

Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828.

Blum, A. and Rivest, R. (1988). Training a 3-node neural network is np-complete. *Advances in neural information processing systems*, 1.

Boyd, S. and Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

Charoenphakdee, N., Cui, Z., Zhang, Y., and Sugiyama, M. (2021). Classification with rejection based on cost-sensitive classification. In *International Conference on Machine Learning*, pages 1507–1517. PMLR.

Charusaie, M.-A., Mozannar, H., Sontag, D., and Samadi, S. (2022). Sample efficient learning of predictors that complement humans. In *International Conference on Machine Learning*, pages 2972–3005. PMLR.

Chow, C. (1970). On optimum recognition error and reject tradeoff. *IEEE Transactions on information theory*, 16(1):41–46.

Cortes, C., DeSalvo, G., and Mohri, M. (2016). Learning with rejection. In *International Conference on Algorithmic Learning Theory*, pages 67–82. Springer.

Davidson, T., Warmsley, D., Macy, M., and Weber, I. (2017). Automated hate speech detection and the problem of offensive language. In *Eleventh international aaai conference on web and social media*.

De, A., Koley, P., Ganguly, N., and Gomez-Rodriguez, M. (2020). Regression under human assistance. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 2611–2620.

Dressel, J. and Farid, H. (2018). The accuracy, fairness, and limits of predicting recidivism. *Science advances*, 4(1):eaao5580.

El-Yaniv, R. and Wiener, Y. (2010). On the foundations of noise-free selective classification. *Journal of Machine Learning Research*, 11(May):1605–1641.

Gangrade, A., Kag, A., and Saligrama, V. (2021). Selective classification via one-sided prediction. In *International Conference on Artificial Intelligence and Statistics*, pages 2179–2187. PMLR.

Geifman, Y. and El-Yaniv, R. (2017). Selective classification for deep neural networks. In *Advances in neural information processing systems*, pages 4878–4887.

Gurobi Optimization, LLC (2022). Gurobi Optimizer Reference Manual.

Guruswami, V. and Raghavendra, P. (2009). Hardness of learning halfspaces with noise. *SIAM Journal on Computing*, 39(2):742–765.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708.

Jacobs, M., Pradier, M. F., McCoy, T. H., Perlis, R. H., Doshi-Velez, F., and Gajos, K. Z. (2021). How machine-learning recommendations influence clinician treatment selections: the example of antidepressant selection. *Translational psychiatry*, 11(1):1–9.

Kakade, S. and Tewari, A. (2008). Rademacher composition and linear prediction. https://home.ttic.edu/~tewari/lectures/lecture17.pdf.

Kerrigan, G., Smyth, P., and Steyvers, M. (2021). Combining human predictions with model probabilities via confusion matrices and calibration. *Advances in Neural Information Processing Systems*, 34.

Keswani, V., Lease, M., and Kenthapadi, K. (2021). Towards unbiased and accurate deferral to multiple experts. *arXiv preprint arXiv:2102.13004*.

Khot, S. and Saket, R. (2011). On the hardness of learning intersections of two halfspaces. *Journal of Computer and System Sciences*, 77(1):129–141.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images. *Citeseer*.

Kumar, A., Raghunathan, A., Jones, R. M., Ma, T., and Liang, P. (2022). Fine-tuning can distort pretrained features and underperform out-of-distribution. In *International Conference on Learning Representations*.

Lin, Y. (2002). Support vector machines and the bayes rule in classification. *Data Mining and Knowledge Discovery*, 6(3):259–275.

Liu, H., Lai, V., and Tan, C. (2021a). Understanding the effect of out-of-distribution examples and interactive explanations on human-ai decision making. *Proceedings of the ACM on Human-Computer Interaction*, 5(CSCW2):1–45.

Liu, J., Gallego, B., and Barbieri, S. (2021b). Incorporating uncertainty in learning to defer algorithms for safe computer-aided diagnosis. *arXiv preprint arXiv:2108.07392*.

Long, P. and Servedio, R. (2013). Consistency versus realizable h-consistency for multiclass classification. In *International Conference on Machine Learning*, pages 801–809. PMLR.

Loshchilov, I. and Hutter, F. (2017). Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Madras, D., Pitassi, T., and Zemel, R. (2018). Predict responsibly: Improving fairness and accuracy by learning to defer. In *Advances in Neural Information Processing Systems*, pages 6150–6160.

Majkowska, A., Mittal, S., Steiner, D. F., Reicher, J. J., McKinney, S. M., Duggan, G. E., Eswaran, K., Cameron Chen, P.-H., Liu, Y., Kalidindi, S. R., et al. (2020). Chest radiograph interpretation with deep learning models: assessment with radiologist-adjudicated reference standards and population-adjusted evaluation. *Radiology*, 294(2):421–431.

Mozannar, H., Satyanarayan, A., and Sontag, D. (2022). Teaching humans when to defer to a classifier via exemplars. In *Proceedings of the Thirty-Sixth AAAI Conference on Artificial Intelligence (AAAI)*.

Mozannar, H. and Sontag, D. (2020). Consistent estimators for learning to defer to an expert. In *International Conference on Machine Learning*, pages 7076–7087. PMLR.

Nguyen, T. and Sanner, S. (2013). Algorithms for direct 0–1 loss optimization in binary classification. In *International Conference on Machine Learning*, pages 1085–1093. PMLR.

Okati, N., De, A., and Gomez-Rodriguez, M. (2021). Differentiable learning under triage. *arXiv preprint arXiv:2103.08902*.

Pradier, M. F., Zazo, J., Parbhoo, S., Perlis, R. H., Zazzi, M., and Doshi-Velez, F. (2021). Preferential mixture-of-experts: Interpretable models that rely on human expertise as much as possible. *arXiv preprint arXiv:2101.05360*.

Raghu, M., Blumer, K., Corrado, G., Kleinberg, J., Obermeyer, Z., and Mullainathan, S. (2019). The algorithmic automation problem: Prediction, triage, and human effort. *arXiv preprint arXiv:1903.12220*.

Raman, N. and Yee, M. (2021). Improving learning-to-defer algorithms through fine-tuning. *arXiv preprint arXiv:2112.10768*.

Razavian, N., Blecker, S., Schmidt, A. M., Smith-McLallen, A., Nigam, S., and Sontag, D. (2015). Population-level prediction of type 2 diabetes from claims data and analysis of risk factors. *Big Data*, 3(4):277–287.

Reimers, N. and Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Ustun, B. and Rudin, C. (2016). Supersparse linear integer models for optimized medical scoring systems. *Machine Learning*, 102(3):349–391.

Verma, R. and Nalisnick, E. (2022). Calibrated learning to defer with one-vs-all classifiers. *arXiv preprint arXiv:2202.03673*.

Wang, X., Peng, Y., Lu, L., Lu, Z., Bagheri, M., and Summers, R. (2017). Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In *IEEE CVPR*, volume 7.

Wilder, B., Horvitz, E., and Kamar, E. (2020). Learning to complement humans. *arXiv preprint arXiv:2005.00582*.

Zagoruyko, S. and Komodakis, N. (2016). Wide residual networks. *arXiv preprint arXiv:1605.07146*.

Zhang, M. and Agarwal, S. (2020). Bayes consistency vs. h-consistency: The interplay between surrogate loss functions and the scoring function class. *Advances in neural information processing systems*, 33:16927–16936.

A Practitioner's guide to our approach

A.1 MILP

We implement the MILP (8)-(13) in the binary setting using the Gurobi Optimizer Gurobi Optimization, LLC (2022) in Python.

```
class MILPDefer:
    def __init__(self, n_classes, time_limit=-1, add_regularization=False,
                 lambda_reg=1, verbose=False):
        self.n_classes = n_classes
        self.time_limit = time_limit
        self.verbose = verbose
        self.add_regularization = add_regularization
        self.lambda_reg = lambda_reg

    def fit(self, dataloader_train, dataloader_val, dataloader_test):
        self.fit_binary(dataloader_train, dataloader_val, dataloader_test)

    def fit_binary(self, dataloader_train, dataloader_val, dataloader_test):
        data_x = dataloader_train.dataset.tensors[0]
        data_y = dataloader_train.dataset.tensors[1]
        human_predictions = dataloader_train.dataset.tensors[2]

        C = 1
        gamma = 0.00001
        Mi = C + gamma
        Ki = C + gamma
        max_data = len(data_x)
        hum_preds = 2*np.array(human_predictions) - 1
        # add extra dimension to x
        data_x_original = torch.clone(data_x)
        norm_scale = max(torch.norm(data_x_original, p=1, dim=1))
        last_time = time.time()
        # normalize data_x and then add dimension
        data_x = torch.cat((torch.ones((len(data_x)), 1),
                            data_x/norm_scale), dim=1).numpy()
        data_y = 2*data_y - 1  # covert to 1, -1
        max_data = max_data  # len(data_x)
        dimension = data_x.shape[1]

        model = gp.Model("milp_deferral")
        model.Params.IntFeasTol = 1e-9
        model.Params.MIPFocus = 0
        if self.time_limit != -1:
            model.Params.TimeLimit = self.time_limit

        H = model.addVars(dimension, lb=[-C] *
                          dimension, ub=[C]*dimension, name="H")
        Hnorm = model.addVars(
            dimension, lb=[0]*dimension, ub=[C]*dimension, name="Hnorm")
        Rnorm = model.addVars(
            dimension, lb=[0]*dimension, ub=[C]*dimension, name="Rnorm")
        R = model.addVars(dimension, lb=[-C] *
                          dimension, ub=[C]*dimension, name="R")
        phii = model.addVars(max_data, vtype=gp.GRB.CONTINUOUS, lb=0)
        psii = model.addVars(max_data, vtype=gp.GRB.BINARY)
        ri = model.addVars(max_data, vtype=gp.GRB.BINARY)

        equal = np.array(data_y) == hum_preds * 1.0
        human_err = 1-equal

        if self.add_regularization:
            model.setObjective(gp.quicksum([phii[i] + ri[i]*human_err[i]
            for i in range(max_data)])/max_data + self.lambda_reg * gp.quicksum(
                [Hnorm[j] for j in range(dimension)])
```

```
                + self.lambda_reg * gp.quicksum([Rnorm[j] for j in range(dimension)]))
        else:
            model.setObjective(gp.quicksum(
                [phii[i] + ri[i]*human_err[i] for i in range(max_data)])/max_data)
        for i in range(max_data):
            model.addConstr(phii[i] >= psii[i] - ri[i], name="phii" + str(i))
            model.addConstr(Mi*psii[i] >= gamma - data_y[i]*gp.quicksum(
                H[j] * data_x[i][j] for j in range(dimension)), name="psii" + str(i))
            model.addConstr(gp.quicksum([R[j]*data_x[i][j] for j in range(dimension)]) >=
            Ki*( ri[i]-1) + gamma*ri[i], name="Riub" + str(i))
            model.addConstr(gp.quicksum([R[j]*data_x[i][j] for j in range(
                dimension)]) <= Ki*ri[i] + gamma*(ri[i]-1),  name="Rilb" + str(i))
            model.update()
        if self.add_regularization:
            for j in range(dimension):
                model.addConstr(Hnorm[j] >= H[j], name="Hnorm1" + str(j))
                model.addConstr(Hnorm[j] >= -H[j], name="Hnorm2" + str(j))
                model.addConstr(Rnorm[j] >= R[j], name="Rnorm1" + str(j))
                model.addConstr(Rnorm[j] >= -R[j], name="Rnorm2" + str(j))

        model.ModelSense = 1  # minimize
        model._time = time.time()
        model._time0 = time.time()
        model._cur_obj = float('inf')
        # model.write('model.lp')
        if self.verbose:
            model.optimize()
        else:
            model.optimize()
        # check if halspace solution has 0 error
        error_v = 0
        rejs = 0
        for i in range(max_data):
            rej_raw = np.sum([R[j].X * data_x[i][j] for j in range(dimension)])
            pred_raw = np.sum([H[j].X * data_x[i][j]
                               for j in range(dimension)])
            if rej_raw > 0:
                rejs += 1
                error_v += (data_y[i] * hum_preds[i] != 1)
            else:
                pred = (pred_raw > 0)
                error_v += (data_y[i] != (2*pred-1))

        self.H = [H[j].X for j in range(dimension)]
        self.R = [R[j].X for j in range(dimension)]
        self.run_time = model.Runtime
        self.norm_scale = norm_scale
        self.train_error = error_v/max_data
```

A.2 Realizable Surrogate

We implement the `RealizableSurrogate` in PyTorch. We showcase the loss function L_{RS} below:

```
def realizable_surrogate_loss(outputs, human_is_correct, labels, lambdaa):
    '''
    outputs (tensor): outputs of model with K+1 output heads (without softmax)
    human_is_correct (tensor): binary tensor indicating if human is
                    correct on each point I_{h=y}
    labels (tensor): list of targets y_i
    lambdaa (float in [0,1]): trade-off parameter in loss

    return: loss (single tensor)
    '''
    batch_size = outputs.size()[0]
    outputs_exp = torch.exp(outputs)
```

```
    rs_loss =  -torch.log2(( m * outputs_exp[range(batch_size), -1]
    + outputs_exp[range(batch_size),labels] )  /(torch.sum(outputs_exp, dim = 1) +eps_cst
                                                ))
    ce_loss = -torch.log2(( outputs_exp[range(batch_size),labels] )
            /(torch.sum(outputs_exp[range(batch_size),:-1], dim = 1)  +eps_cst ))
    loss = lambdaa*rs_loss + (1-lambdaa)*ce_loss
    return torch.sum(loss)/batch_size
```

B MILP

B.1 Verification

The MILP in the binary setting is formulated as:

$$M^*, R^*, . = \arg \min_{M,R,\{r_i\},\{t_i\},\{\phi_i\}} \sum_i \phi_i + r_i \mathbb{I}_{h_i \neq y_i} \tag{15}$$

$$\phi_i \geq t_i - r_i, \qquad \phi_i \geq 0 \quad \forall i \in [n] \tag{16}$$

$$K_m t_i \geq \gamma_h - y_i M^\top x_i \quad \forall i \in [n] \tag{17}$$

$$R^\top x_i \leq K_r r_i + \gamma_r (r_i - 1), \quad R^\top x_i \geq K_r (r_i - 1) + \gamma_r r_i \quad \forall i \in [n] \tag{18}$$

$$-C \leq R_i \leq C, \quad -C \leq M_i \leq C \quad \forall i \in [d] \tag{19}$$

$$r_i \in \{0,1\}, t_i \in \{0,1\}, \phi_i \in \mathbb{R}^+ \quad \forall i \in [n],\ R, M \in \mathbb{R}^d \tag{20}$$

Extension to Multiclass. The above MILP only applies to binary labels but we can generalize it to the multiclass setting where $\mathcal{Y} = \{1, \cdots, C\}$. In this case, we have a coefficient vector M_j for each class $j \in \mathcal{Y}$, and $m(x) = \arg\max_{j \in \mathcal{Y}} M_j^\top x$. Given a labeled point (x, y), we let $c_j = \text{sign}(M_y^\top x - M_j^\top x)$ for $j \neq y$, and let $t_i = \mathbb{I}_{\sum_{j \neq y} c_j < C-1}$. Then if $m(x) = y$, we must have $c_j = 1$ for all $j \neq y$ and thus $t_i = 0$ which means that the classifier is correct. Similarly, if there exists a $j \neq y$ for which $c_j = -1$, it means the classifier is incorrect and accordingly $t_i = 1$. We can reformulate these indicator constraints using a similar big-M approach as above. The formulation is below:

$$M^*, R^*, . = \arg \min_{M,R,\{r_i\},\{t_i\},\{c_{ij}\},\{\phi_i\}} \sum_i \phi_i + r_i \mathbb{I}_{h_i \neq y_i} \tag{21}$$

$$\phi_i \geq t_i - r_i, \qquad \phi_i \geq 0 \quad \forall i \in [n] \tag{22}$$

$$(M_{y_i} - M_j)^\top x_i \leq 2K_h c_{ij} + \gamma_h (c_{ij} - 1),$$
$$(M_{y_i} - M_j)^\top x_i \geq 2K_h (c_{ij} - 1) + \gamma_h c_{ij} \quad \forall i \in [n]\, \forall j \in [C] \neq y_i \tag{23}$$

$$t_i \geq (C - 1 - \sum_{j \in [L], j! = y_i} c_{ij})/(C-1) \tag{24}$$

$$R^\top x_i \leq K_r r_i + \gamma_r (r_i - 1), \quad R^\top x_i \geq K_r (r_i - 1) + \gamma_r r_i \quad \forall i \in [n] \tag{25}$$

$$-C \leq R_i \leq C, \quad -C \leq M[i,l] \leq C \quad \forall i \in [d]\, \forall l \in [C] \tag{26}$$

$$r_i \in \{0,1\}, t_i \in \{0,1\}, c_{ij} \in \{0,1\}, \phi_i \in \mathbb{R}^+ \quad \forall i \in [n],\ R, M \in \mathbb{R}^d \tag{27}$$

Let us verify the formulations above.

The variable $\phi_i \geq \max(t_i - r_i, 0)$, the RHS takes values either 0 or 1, since ϕ_i in the objective then the optimal value is either 0 or 1 as well so that $\phi_i = \max(t_i - r_i, 0) = (1 - r_i) t_i$.

For t_i in the binary case: when $y_i M^\top x_i$ is positive, then $\gamma_h - y_i M^\top x_i$ is negative since $|M^\top x_i| \geq \gamma_h$ by Assumption 2, so that to satisfy constraint (17) either value of 0 or 1 are valid for t_i, however since t_i shows up in the objective then the optimal value is 0. On the other hand, when $y_i M^\top x_i$ is negative, then $\gamma_h - y_i M^\top x_i$ is positive, so that the only valid option for t_i is 1 and since $M^\top x_i \leq K_m$ then the constraint can be satisfied. So that we proved that $t_i = sign(y_i M^\top x_i)$.

We previously verified constraint for r_i and R in the body. When $r_i = 0$ then we have the constraints $R^\top x_i \leq -\gamma_r$ and $R^\top x_i \geq -K_r$: this forces the rejector to be negative which is consistent. When $r_i = 1$, we have $R^\top x_i \geq \gamma_r$ and $R^\top x_i \leq K_r$: which means the rejector is positive. Thus we proved $r_i = \mathbb{I}(R^\top x_i \geq 0)$.

For t_i in the multiclass settings: by analogy to the constraints for R and r_i it is easy to see that the variable $c_{ij} = sign(H_{y_i}^\top x_i - H_j^\top x_i)$. For a given x_i, y_i, the classification is only correct if $c_{ij} = 1$ for all $j \in [C] \neq y_i$ so that $\arg\max_j H_i^\top x_i = y_i$. We can then see that we set $t_i = \mathbb{I}(\sum_{j \neq y_i} c_{ij}/(C-1) \neq 1)$ so that t_i denotes the error of our classifier on example i.

C Experimental Details and Results

C.1 Baseline Implementation

OvASurrogate (Verma and Nalisnick, 2022): We rely on the loss implementation available online at [2].

DifferentiableTriage (Okati et al., 2021): We rely on the implementation in [3]. Note that the differentiable triage method implementation in Okati et al. (2021) relies on having loss estimates of the human, particularly cross entropy loss estimates, which requires the conditional probabilities $\mathbb{P}(H = i|X = x)$ for each $i \in \mathcal{Y}$. However, in our setting, we only have samples of the human decisions m_i, not probabilistic estimates. The method can be summarized as a two-stage method: 1) classifier training: at each epoch, only train on points where classifier loss is lower than human loss, 2) rejector training: fit the rejector to predict who between the classifier and the human has lower loss. Since we only have samples of human behavior, we use the $0-1$ loss of the classifier and the human on an example basis for comparison.

CrossEntropySurrogate (Mozannar and Sontag, 2020): We rely on the implementation in [4]. We tune the parameter α over the grid $[0, 0.1, 0.5, 1]$ on the validation set.

CompareConfidence (Raghu et al., 2019): we train the classifier using the cross entropy loss on all the data, we then train a model to predict if the human is correct or not on each example in the training set. For each test point, we compare the confidence of the classifier versus the human correctness model and defer accordingly.

SelectivePrediction: we train the classifier using the cross entropy loss on all the data, for the rejector, we learn a single threshold on the validation set for the classifier confidence (probability of the predicted class) in order to maximize system accuracy.

C.2 Training Details

Table 2: Training details for each dataset, we use the Adam optimizer (Kingma and Ba, 2014) and AdamW (Loshchilov and Hutter, 2017)

Dataset	Optimizer	Number of Epochs	Learning Rate
SyntheticData (ours)	Adam	300	0.1
CIFAR-K	Adam	100	0.001
CIFAR-10H (Battleday et al., 2020)	AdamW	20	0.001
Imagenet-16H (Kerrigan et al., 2021)	Adam	20	0.001
HateSpeech (Davidson et al., 2017)	Adam	50	0.001
COMPASS (Dressel and Farid, 2018)	Adam	300	0.1
NIH Chest X-ray (Wang et al., 2017; Majkowska et al., 2020)	AdamW	3	0.001

C.3 Synthetic Data

Figure 3a in the main body shows the sample complexity of the different methods on the uniform data distribution. We show in Figure **??** the performance of the different methods with the same setup with the uniform data distribution.

[2] https://github.com/rajevv/OvA-L2D
[3] https://github.com/Networks-Learning/differentiable-learning-under-triage
[4] https://github.com/clinicalml/learn-to-defer

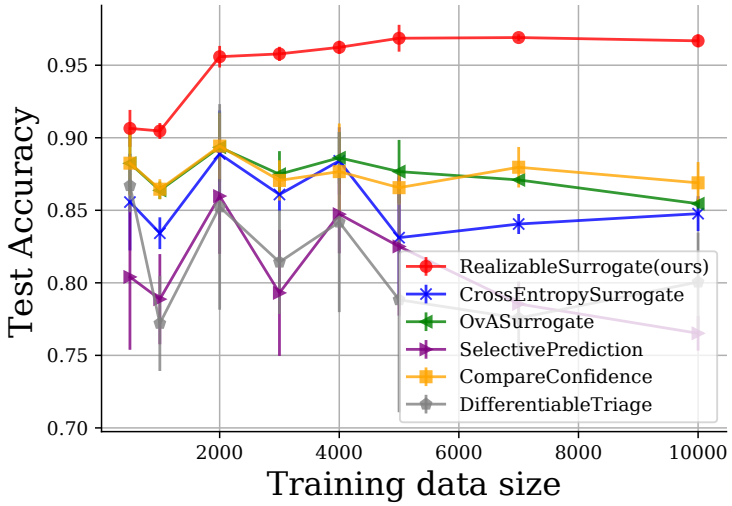


Figure 6: (Test performance of the different methods on realizable synthetic data as we increase the training data size and repeat the randomization over 10 trials to get standard errors on uniform data.

We also experiment with making the data unrealizable by setting ($d = 10$, $p_m = 0.1, p_{h0} = 0.4, p_{h1} = 0.1$, Gaussian distribution with 20 clusters) in Figure 7.

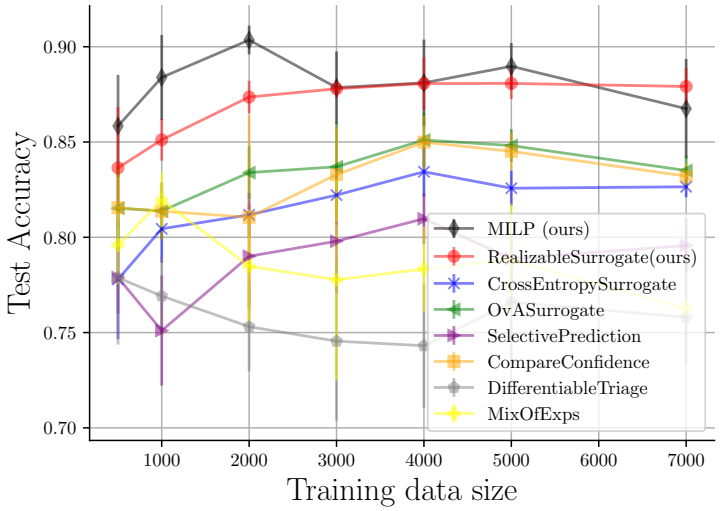
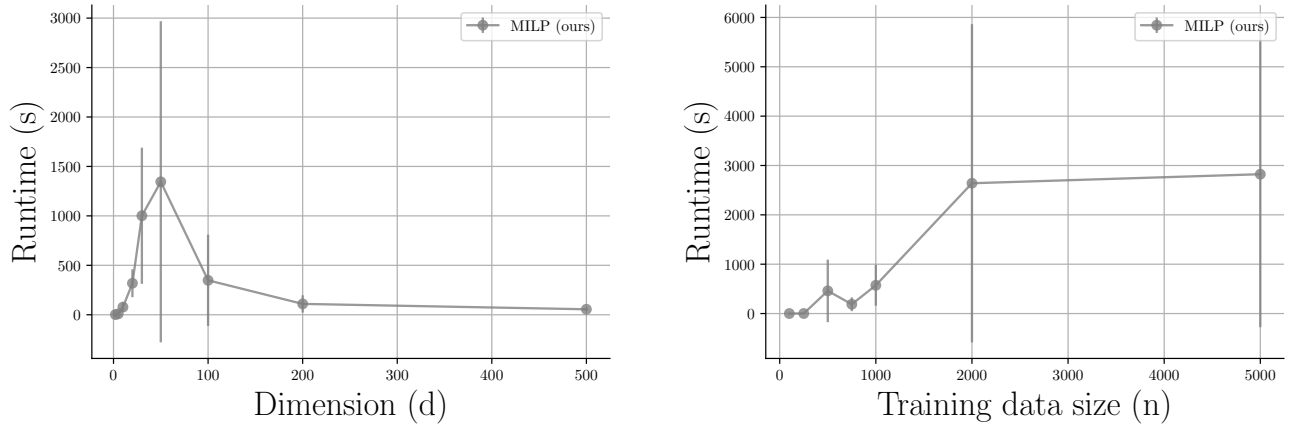


Figure 7: (Test performance of the different methods on unrealizable ($d = 10$, $p_m = 0.1, p_{h0} = 0.4, p_{h1} = 0.1$, Gaussian distribution with 20 clusters) synthetic data as we increase the training data size.

We also show average run-times for the MILP on the synthetic data as we increase the dimension in Figure 8a and as we increase the training data size in Figure 8b. The distribution was uniform and realizable with $p_m = 0.0, p_{h0} = 0.3, p_{h1} = 0.0$. We observe that the run time increases with training set size which is the biggest bottleneck. The runtime also increases with dimension up until the dimension is of the same order as the number of training points, afterwards it is faster for the MILP to find a 0 error solution.

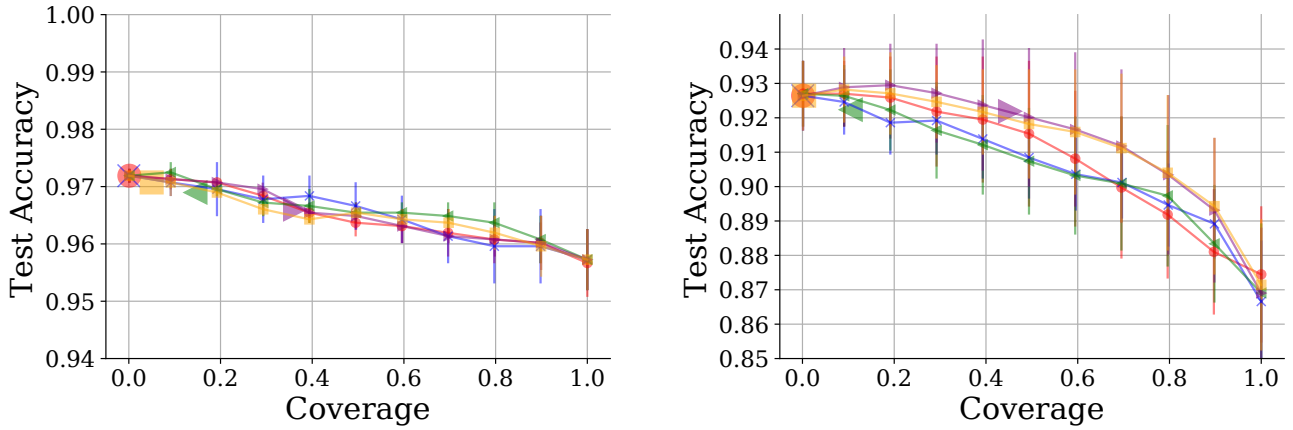


(a) Runtime with increasing dimension, $n = 1000$

(b) Runtime with increasing training data size, $d = 30$

Figure 8: Runtime of the MILP on the realizable synthetic data with uniform data distribution. Note that the test accuracy of the MILP is demonstrated in Figure 3a and the MILP always reaches 0 training error across the different data dimensions and training set sizes.

C.4 NIH Chest X-ray



(a) Fracture

(b) Nodule or Mass

Figure 9: NIH Chest X-ray results on the two remaining tasks with the baselines and our method and red with circle markers. We see that all methods aren't able to obtain a performance of a human-AI team with better performance than the human, our method on both tasks defers to the human.

D Deferred Proofs and Derivations

D.1 Related Work

We mentioned that the surrogate in Verma and Nalisnick (2022) belongs to the family derived in Charusaie et al. (2022). This is established by setting $l_\phi(i, f(x))$ as follows [5]:

$$l_\phi(i, f(x)) = \begin{cases} \phi(g_y) + \sum_{y' \neq y} \phi(-g_{y'}), & if\ y \in \mathcal{Y} \\ \phi(g_y) - \phi(-g_{y'}), \end{cases} \tag{28}$$

[5] This was established by Yuzhou Cao.

D.2 Section 4 (Hardness)

D.2.1 Background and Definitions

Realizable Intersection of Halfspaces. For our purposes, an instance $\mathcal{I}$ of learning an intersection of halfspaces in the realizable setting is given by a finite dataset $\{(x_i, y_i)\}_{i=1}^n$, with $x_i \in \mathbb{R}^d$ and $y_i \in \{0, 1\}$, such that there exist halfspaces $g_1^* : \mathbb{R}^d \to \{0, 1\}$ and $g_2^* : \mathbb{R}^d \to \{0, 1\}$ with zero error on the dataset:

$$err_{\mathcal{I}}(g_1^*, g_2^*) := \frac{1}{n} \sum_i \mathbb{I}_{g_1^*(x_i) \wedge g_2^*(x_i) \neq y_i} = 0.$$

We consider two related problems: finding halfspaces (g_1, g_2) with *exact* and *weak* agreement.

Exact agreement. Given an instance $\mathcal{I}$ of realizable intersection of halfspaces, the exact agreement problem is to find a pair of halfspaces (g_1, g_2) such that $g_1(x_i) \wedge g_2(x_i) = y_i$ for all $i \in \{1, \ldots, n\}$.

Weak agreement. Given an instance $\mathcal{I}$ of realizable intersection of halfspaces, the weak agreement problem is to find a pair of halfspaces (g_1, g_2) with error at most $1/2 - \gamma$ for some $\gamma > 0$:

$$err_{\mathcal{I}}(g_1, g_2) := \frac{1}{n} \sum_i \mathbb{I}_{g_1(x_i) \wedge g_2(x_i) \neq y_i} \leq \frac{1}{2} - \gamma.$$

Note that there exists a pair (g_1^*, g_2^*) with error 0 but the goal is just to obtain error $1/2 - \gamma$.

Quite a bit is known about the hardness of the exact and weak agreement problems.

Theorem (Blum and Rivest (1988) Theorem 1, rephrased)**.** *The exact agreement problem is NP-hard.*

Theorem (Khot and Saket (2011) Theorem 2, rephrased)**.** *There is no polynomial-time algorithm for the weak agreement problem unless $NP = RP$.*

We also consider finite-data versions of LWD-H:

Finite-data realizable LWD-H. An instance $\mathcal{J}$ of learning with deferral in the realizable setting is given by a finite dataset $\{(x_i, y_i, h_i)\}_{i=1}^n$, with $x_i \in \mathbb{R}^d$ and $y_i, h_i \in \{0, 1\}$, such that there exist halfspaces $m^* : \mathbb{R}^d \to \{0, 1\}$ and $r^* : \mathbb{R}^d \to \{0, 1\}$ with zero error on the dataset:

$$err_{\mathcal{J}}(m^*, r^*) := \frac{1}{n} \sum_i \mathbb{I}_{r^*(x_i)=1} \mathbb{I}_{h_i \neq y_i} + \mathbb{I}_{r^*(x_i)=0} \mathbb{I}_{m^*(x_i) \neq y_i} = 0.$$

As with intersection-of-halfspaces, we can consider finding halfspace classifier/rejector pairs (m, r) with *exact* and *weak* agreement.

Exact agreement. Given an instance $\mathcal{J}$ of realizable LWD-H, the exact agreement problem is to find a pair of halfspaces (m, r) such that for all i, if $r(x_i) = 0$, $m(x_i) = y_i$, and if $r(x_i) = 1$, $h_i = y_i$. That is, the error of the classifier/human system on the finite dataset is 0.

Weak agreement. Given an instance $\mathcal{J}$ of realizable LWD-H, the weak agreement problem is to find a pair of halfspaces (m, r) with error at most $1/2 - \gamma$ for some $\gamma > 0$:

$$err_{\mathcal{J}}(m, r) := \frac{1}{n} \sum_i \mathbb{I}_{r(x_i)=1} \mathbb{I}_{h_i \neq y_i} + \mathbb{I}_{r(x_i)=0} \mathbb{I}_{m(x_i) \neq y_i} \leq \frac{1}{2} - \gamma.$$

D.2.2 Mapping between learning intersections and LWD-H

We show how to turn an instance $\mathcal{I}$ of realizable intersection of halfspaces into an instance of $\mathcal{J}$ of (finite-data) realizable LWD-H. Given an arbitrary instance $\mathcal{I}$ on dataset $\mathcal{D}$, Lemma D.1 shows how to construct an instance $\mathcal{J}$ of LWD-H and a bijection $(g_1, g_2) \longleftrightarrow (m, r)$ such that for arbitrary halfspaces (g_1, g_2), the error $err_{\mathcal{I}}(g_1, g_2) = err_{\mathcal{J}}(m, r)$. In particular, since we assumed $\mathcal{I}$ is realizable and hence $\exists g_1^*, g_2^*$ with $err_{\mathcal{I}}(g_1^*, g_2^*) = 0$, Lemma D.1 shows how to construct an instance

$\mathcal{J}$ of LWD-H with $err_{\mathcal{J}}(m^*, r^*) = 0$. This will allow us to reduce an arbitrary instance $\mathcal{I}$ of realizable intersection of halfspaces to an instance $\mathcal{J}$ of realizable LWD-H. Additionally, given an arbitrary classifier/rejector pair (m, r) on this $\mathcal{J}$ with error ϵ, Lemma D.1 shows how to map $(m, r) \to (g_1, g_2)$ with error ϵ on instance $\mathcal{I}$.

Lemma D.1. *Consider an arbitrary instance $\mathcal{I}$ of learning an intersection of halfspaces on a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$. Define $\widetilde{\mathcal{D}} = \{(x_i, y_i, 0)\}_{i=1}^n$. This corresponds to an instance $\mathcal{J}$ of LWD-H where the "human expert" always outputs label 0.*

Then:

1. *Consider two arbitrary halfspaces g_1, g_2 and set $m(x) = g_1(x)$, $r(x) = 1 - g_2(x)$. Note that m and r are also halfspaces. Then $err_{\mathcal{I}}(g_1, g_2) = err_{\mathcal{J}}(m, r)$. That is,*

$$\frac{1}{n} \sum_{(x_i, y_i) \in \mathcal{D}} \mathbb{I}[g_1(x_i) \wedge g_2(x_i) \neq y_i] = \frac{1}{n} \sum_{(x_i, y_i, h_i) \in \widetilde{\mathcal{D}}} \left(\mathbb{I}_{r(x_i)=1} \mathbb{I}_{h_i \neq y_i} + \mathbb{I}_{r(x_i)=0} \mathbb{I}_{m(x_i) \neq y_i} \right).$$

2. *Suppose $\mathcal{I}$ is an instance of* realizable *intersection of halfspaces. Then the instance $\mathcal{J}$ of LWD-H defined by the dataset $\widetilde{\mathcal{D}}$ is an instance of* realizable *LWD-H. That is, there exists (m^*, r^*) with $err_{\mathcal{J}}(m^*, r^*) = 0$.*

Proof. For part 1, recall that by definition:

$$\text{err}_{\mathcal{J}}(m, r) = \frac{1}{n} \sum_{(x_i, h_i, y_i) \in \widetilde{\mathcal{D}}} \left(\mathbb{I}_{r(x_i)=1} \mathbb{I}_{h_i \neq y_i} + \mathbb{I}_{r(x_i)=0} \mathbb{I}_{m(x_i) \neq y_i} \right).$$

Since $h_i = 0$ for all i, this is equal to

$$\frac{1}{n} \sum_i \left(\mathbb{I}_{r(x_i)=1} \mathbb{I}_{y_i=1} + \mathbb{I}_{r(x_i)=0} \mathbb{I}_{m(x_i) \neq y_i} \right).$$

Using $r(x) = 1 - g_2(x)$ and $m(x) = g_1(x)$, this simplifies further to:

$$\frac{1}{n} \sum_i \left(\mathbb{I}_{g_2(x_i)=0} \mathbb{I}_{y_i=1} + \mathbb{I}_{g_2(x_i)=1} \mathbb{I}_{g_1(x_i) \neq y_i} \right). \tag{29}$$

Consider the error of $err_{\mathcal{I}}(g_1, g_2)$. The model makes a mistake if $g_2(x) = 0$ and $y(x) = 1$, $g_2(x) = g_1(x) = 1$ and $y = 0$, or $g_2(x) = 1, g_1(x) = 0$, and $y = 1$. The first case is $\mathbb{I}_{g_2(x)=0} \mathbb{I}_{y=1}$ and the latter two cases can be expressed as $\mathbb{I}_{g_2(x)=1} \mathbb{I}_{g_1(x) \neq y}$. Hence

$$err_{\mathcal{I}}(g_1, g_2) = \frac{1}{n} \sum_{(x_i, y_i) \in \widetilde{\mathcal{D}}} \mathbb{I}[g_1(x_i) \wedge g_2(x_i) \neq y_i] = \frac{1}{n} \sum_i \left(\mathbb{I}_{g_2(x_i)=0} \mathbb{I}_{y_i=1} + \mathbb{I}_{g_2(x_i)=1} \mathbb{I}_{g_1(x_i) \neq y_i} \right),$$

which is equal to (29), so $err_{\mathcal{I}}(g_1, g_2) = \text{err}_{\mathcal{J}}(m, r)$.

For part 2, we assumed that $\mathcal{I}$ was realizable, so there exists g_1^*, g_2^* with $err_{\mathcal{I}}(g_1^*, g_2^*) = 0$. Applying part 1 yields m^*, r^* such that $err_{\mathcal{J}}(m^*, r^*) = 0$. Hence $\mathcal{J}$ is an instance of realizable LWD-H. □

Lemma D.1 takes an instance $\mathcal{I}$ of learning an intersection of halfspaces and constructs an instance $\mathcal{J}$ of LWD-H such that there is an error-preserving bijection between solutions of $\mathcal{I}$ and solutions of $\mathcal{J}$. This allows us to easily apply the existing hardness results for learning a realizable intersection of halfspaces, since if $\mathcal{I}$ is realizable then so is $\mathcal{J}$.

D.2.3 Hardness results for LWD-H

Theorem D.1. *There is no polynomial-time algorithm for solving the exact agreement problem for LWD-H unless P=NP.*

Proof. Suppose there exists a polytime algorithm $\mathcal{A}$ for solving exact agreement on realizable LWD-H. Consider an arbitrary instance $\mathcal{I}$ of learning a realizable intersection of halfspaces. Lemma D.1 shows how to construct an instance $\mathcal{J}$ of realizable LWD-H. Run Algorithm $\mathcal{A}$ on $\mathcal{J}$ to obtain halfspaces (m, r) with $err_{\mathcal{J}}(m, r) = 0$. Set $g_1 = m$, $g_2 = 1 - r$. Lemma D.1 guarantees that $err_{\mathcal{I}}(g_1, g_2) = 0$. Hence, $\mathcal{A}$ is a polynomial-time algorithm for exact agreement for realizable intersection of halfspaces. Blum and Rivest (1988) shows that there is no polynomial-time algorithm for exact agreement for realizable intersection of halfspaces unless $P = NP$. □

Corollary D.1. *There is no efficient, proper PAC learner for realizable LWD-H unless $NP = RP$.*

Proof sketch. Suppose $\mathcal{A}$ is an efficient proper PAC learner for realizable LWD-H, so for any distribution $\mathcal{D}$, any $\epsilon > 0$, $\delta > 0$, given $poly(1/\delta, 1/\epsilon)$ samples from $\mathcal{D}$, $\mathcal{A}$ outputs a pair of halfspaces (m, r) with (population) system error at most ϵ in time $poly(1/\epsilon, 1/\delta)$.

Now let $\mathcal{D}$ be the uniform distribution over a dataset of n points $\{(x_i, y_i, h_i)\}_{i=1}^n$. Set $\epsilon = 1/(2n)$ and $\delta = 1/100$ and run $\mathcal{A}$. With probability at least $1 - \delta$ $\mathcal{A}$ outputs (m, r) with error at most $1/(2n)$. Of course, if (m, r) has error at most $1/(2n)$ it must have error 0. This gives a randomized algorithm for solving the exact agreement problem for realizable finite-data LWD-H. □

These results show that exact agreement, and thus exact proper PAC learning, are hard. Next we consider the hardness of weak agreement.

Theorem 1 *Let $\epsilon > 0$ be an arbitrarily small constant and suppose we have an instance $\mathcal{J}$ of realizable LWD-H. So we have data $\mathcal{D} = \{(x_i, y_i, h_i)\}_{i=1}^n$, where $x_i \in \mathbb{R}^d$, $y_i, h_i \in \{0, 1\}$, and there exist halfspaces m^*, r^* with zero loss on $\mathcal{D}$:*

$$err_{\mathcal{J}}(m^*, r^*) := \frac{1}{n} \sum_i \left(\mathbb{I}_{r^*(x_i)=1} \mathbb{I}_{h_i \neq y_i} + \mathbb{I}_{r^*(x_i)=0} \mathbb{I}_{m^*(x_i) \neq y_i} \right) = 0$$

Then there is no polynomial-time algorithm to find a classifier-rejector pair $(\hat{m}, \hat{r})$ with error $1/2 - \epsilon$, i.e.:

$$\frac{1}{n} \sum_i \left(\mathbb{I}_{\hat{r}(x_i)=1} \mathbb{I}_{h_i \neq y_i} + \mathbb{I}_{\hat{r}(x_i)=0} \mathbb{I}_{\hat{m}(x_i) \neq y_i} \right) \leq \frac{1}{2} - \epsilon$$

unless $NP = RP$.

Proof. Suppose there exists a polynomial-time algorithm $\mathcal{A}$ and a $\gamma > 0$ such that given an instance $\mathcal{J}$ of realizable LWD-H, $\mathcal{A}$ returns a pair $(\hat{m}, \hat{r})$ with error $err_{\mathcal{J}}(\hat{m}, \hat{r})$ at most $1/2 - \gamma$. Consider an arbitrary instance $\mathcal{I}$ of realizable intersection of halfspaces. Lemma D.1 shows how to reduce $\mathcal{I}$ to an instance $\mathcal{J}$ of realizable LWD-H. Run Algorithm $\mathcal{A}$ on $\mathcal{J}$ to obtain a pair of halfspace $(\hat{m}, \hat{r})$ with error at most $err_{\mathcal{J}}(\hat{m}, \hat{r}) \leq 1/2 - \gamma$. Lemma D.1 guarantees that $g_1 = \hat{m}, g_2 = 1 - \hat{r}$ satisfy $err_{\mathcal{I}}(g_1, g_2) \leq 1/2 - \gamma$. Hence $\mathcal{A}$ gives a deterministic algorithm for solving the weak agreement problem for realizable intersection of halfspaces. Khot and Saket (2011, Theorem 4) construct an algorithm/reduction showing that if we can efficiently solve weak agreement for realizable intersection of halfspaces, then Smooth Label Cover is in RP, but Smooth Label Cover is an NP-hard problem (Khot and Saket, 2011, Theorem 3). Hence there is no polynomial-time algorithm to find a classifier-rejector pair $(\hat{m}, \hat{r})$ with error $1/2 - \epsilon$ unless $NP = RP$. □

Corollary D.2. *There is no efficient, proper,* weak *PAC-learner for realizable LWD-H unless $NP = BPP$.*

Proof. Given a distribution $\mathcal{D}$ over points (x, y, h), $x \in \mathbb{R}^d$, $y, h \in \{0, 1\}$ and halfspaces (m, r), let

$$err_{\mathcal{D}}(m, r) := \mathbb{P}_{(x,y,h)\sim\mathcal{D}}[r(x) = 1 \wedge h \neq y \vee r(x) = 0 \wedge m(x) \neq y].$$

This is identical to the *system loss* (2) on distribution $\mathcal{D}$. Suppose there exists an efficient, proper, weak PAC-learner for realizable LWD-H. I.e., there exists some γ such that for any distribution $\mathcal{D}$, under the guarantee that $\exists (m^*, r^*)$ with $err_{\mathcal{D}}(m^*, r^*) = 0$, given access to $poly(1/\delta)$ samples from $\mathcal{D}$, with probability at least $1 - \delta$, $\mathcal{A}$ returns a pair (m, r) with $err_{\mathcal{D}}(m, r) \leq \frac{1}{2} - \gamma$ in $poly(1/\delta)$ time.

By combining Lemma D.1 with the randomized reduction of Khot and Saket (2011), we can use $\mathcal{A}$ to construct an algorithm that implies Smooth Label Cover is in BPP. The definition of Smooth Label Cover is not important for our purposes beyond the following two results:

Theorem D.2. *(Khot and Saket, 2011, Theorem 3) For any constant t and arbitrarily small constants $\mu, \vartheta, \eta > 0$, there exist constants k and m such that given an instance $\mathcal{L}$ of Smooth-Label-Cover$(t, \mu, \vartheta, k, m)$ it is NP-hard to distinguish between the following two cases:*

- *YES Case/Completeness: There is a labeling to the vertices of $\mathcal{L}$ which satisfies all the edges.*
- *NO Case/Soundness: No labeling to the vertices of $\mathcal{L}$ satisfies more than η fraction of the edges.*

Theorem D.3. *(Khot and Saket, 2011, Theorem 4) For any constant $\gamma > 0$ and integer $l > 0$, there is a randomized polynomial time reduction from an instance $\mathcal{L}$ of Smooth-Label-Cover$(t, \mu, \vartheta, k, m)$ to an instance $\mathcal{I}$ of Realizble Intersection of Halfspaces for appropriately chosen parameters (t, μ, ϑ) and soundness η, such that*

- *YES Case/Completeness: If $\mathcal{L}$ is a YES instance, then there is an intersection of two halfspaces which correctly classifies all the points in instance $\mathcal{I}$.*
- *NO Case/Soundness: If $\mathcal{L}$ is a NO instance, then with probability at least 9/10, there is no function of up to l halfspaces that correctly classifies more than $1/2 + \gamma$ fraction of points in instance $\mathcal{I}$.*

For our case, we can use Lemma D.1 to further reduce the instance $\mathcal{I}$ constructed by Theorem D.3 to an instance $\mathcal{J}$ of LWD-H, then run the weak PAC-learner $\mathcal{A}$ on $\mathcal{J}$. If $\mathcal{A}$ outputs a pair of halfspaces (m, r) with error at most $1/2 - \gamma$, we output YES. Otherwise we output NO.

If $\mathcal{I}$ is a realizable instance, $\mathcal{A}$ returns a pair of halfspaces with error at most $1/2 - \gamma$ with probability at least $1 - \delta$. On the other hand, if $\mathcal{I}$ is not weakly realizable (w.r.) (i.e., there is no function of up to l halfspaces that correctly classifies more than a $1/2 + \gamma$ fraction of points in $\mathcal{I}$), then clearly $\mathcal{A}$ never returns a good pair of halfspaces, since no such pair exists. Therefore:

$$\begin{aligned}\mathbb{P}(\text{YES}|\mathcal{L}\text{ YES}) &= \mathbb{P}(\text{YES}|\mathcal{I}\text{ realizable})\mathbb{P}(\mathcal{I}\text{ realizable}|\mathcal{L}\text{ YES})\\ &= (1-\delta)\cdot 1\end{aligned}$$

$$\begin{aligned}\mathbb{P}(\text{NO}|\mathcal{L}\text{ NO}) &= \mathbb{P}(\text{NO}|\mathcal{I}\text{ w.r.})\mathbb{P}(\mathcal{I}\text{ w.r.}|\mathcal{L}\text{ NO}) + \mathbb{P}(\text{NO}|\mathcal{I}\text{ not w.r.})\mathbb{P}(\mathcal{I}\text{ not w.r.}|\mathcal{L}\text{ NO})\\ &\geq \mathbb{P}(\text{NO}|\mathcal{I}\text{ not w.r.})\mathbb{P}(\mathcal{I}\text{ not w.r.}|\mathcal{L}\text{ NO})\\ &\geq \mathbb{P}(\text{NO}|\mathcal{I}\text{ not w.r.})\frac{9}{10}\\ &= \frac{9}{10}.\end{aligned}$$

Hence we can use $\mathcal{A}$ to construct an algorithm for a Smooth-Label-Cover instance $\mathcal{L}$ that outputs YES when $\mathcal{L}$ is a YES with probability at least $(1 - \delta)$, and outputs NO when $\mathcal{L}$ is a NO with probability at least 9/10. Since we assumed $\mathcal{A}$ runs in $poly(1/\delta)$, this implies Smooth Label Cover is in BPP. Together with Theorem D.2, this shows that there is no efficient, proper, weak PAC learner for realizable LWD-H unless $NP = BPP$. □

Finally, we show that when realizability is violated, there is no efficient algorithm for weak agreement.

Corollary 2 (formal). *Let $\delta, \epsilon > 0$ be arbitrarily small constants. Then, given a set of points $\{(x_i, y_i, h_i)\}$ with $x_i \in \mathbb{R}^d$, $y_i, h_i \in \{0, 1\}$ with a guarantee that there is a classifier/rejector pair (m^*, r^*) that classifies a $1 - \delta$ fraction of points correctly, there is no polynomial time algorithm to find a classifier-rejector pair that classifies $\frac{1}{2} + \epsilon$ fraction of points correctly unless P = NP.*

Proof. This is a simple reduction from learning a single halfspace in the presence of noise, which is hard by the following result:

Theorem. *(Guruswami and Raghavendra (2009), see also Khot and Saket (2011, Theorem 1)) Let $\delta, \epsilon > 0$ be arbitrarily small constants. Then, given a set of labeled points $\{(x_i, y_i)\}$ in $\mathbb{R}^d$ with a guarantee that there is a halfspace that classifies $1 - \delta$ fraction of points correctly, there is no polynomial time algorithm to find a halfspace that classifies $1/2 + \epsilon$ fraction of points correctly, unless P = NP.*

Suppose we have an algorithm $\mathcal{A}$ for solving LWD-H in the presence of noise. In particular, there exists some $\epsilon > 0, \delta > 0$ such that under the guarantee that there exists an (m^*, r^*) pair with error at most δ, $\mathcal{A}$ returns an (m, r) pair with error at most $\frac{1}{2} - \epsilon$.

Consider an instance $\mathcal{I}$ of learning a single halfspace in the presence of noise defined by a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$, such that there exists a halfspace c with error at most δ on $\mathcal{D}$. From $\mathcal{D}$, construct the dataset $\widetilde{\mathcal{D}} = \{(x_i, y_i, 1 - y_i)\}_{i=1}^n$. This is an instance $\mathcal{J}$ of LWD-H where the "human expert" is always wrong. Note that $(c, 0)$ is a classifier/rejector pair with error at

most δ on $\widetilde{\mathcal{D}}$, so $\mathcal{J}$ is an instance of LWD-H with noise level δ. Run algorithm $\mathcal{A}$ on $\mathcal{J}$ with parameter ϵ to obtain an (m, r) pair with $err_{\mathcal{J}}(m, r) = 1/2 - \epsilon$. Then:

$$\begin{aligned}
1/2 - \epsilon \geq err_{\mathcal{J}}(m, r) &= \frac{1}{n} \sum_i \mathbb{I}_{r(x_i)=1} \mathbb{I}_{h_i \neq y_i} + \mathbb{I}_{r(x_i)=0} \mathbb{I}_{m(x_i) \neq y_i} \\
&= \frac{1}{n} \left(\sum_{i: r(x_i)=1} \mathbb{I}_{h_i \neq y_i} + \sum_{i: r(x_i)=0} \mathbb{I}_{m(x_i) \neq y_i} \right) \\
&\geq \frac{1}{n} \left(\sum_{i: r(x_i)=1} \mathbb{I}_{m(x_i) \neq y_i} + \sum_{i: r(x_i)=0} \mathbb{I}_{m(x_i) \neq y_i} \right) \\
&= \frac{1}{n} \sum_i \mathbb{I}_{m(x_i) \neq y_i} \\
&= err_{\mathcal{I}}(m),
\end{aligned}$$

where the inequality is because we constructed $\widetilde{\mathcal{D}}$ such that $\mathbb{I}_{h_i \neq y_i} = 1$ for all i. Therefore, there exists a δ and ϵ for which, given a dataset and the guarantee that there exists a halfspace with error at most δ, we can output a halfspace with error at most $1/2 - \epsilon$. Combining this with the Theorem above shows that if $\mathcal{A}$ runs in polynomial time, $P = NP$. □

D.3 Section 5 (MILP)

Proposition 1. *For any expert H and data distribution $\mathbf{P}$ over $\mathcal{X} \times \mathcal{Y}$ that satisfies Assumption 2, let $0 < \delta < \frac{1}{2}$, then with probability at least $1 - \delta$, the following holds for the empirical minimizers $(\hat{m}^*, \hat{r}^*)$ obtained by the MILP:*

$$L^{0-1}_{\text{def}}(\hat{m}^*, \hat{r}^*) \leq \hat{L}^{0-1}_{\text{def}}(\hat{m}^*, \hat{r}^*) \frac{(K_m + K_r) d \sqrt{2 \log d} + 10 \sqrt{\log(2/\delta)}}{\sqrt{n \mathbb{P}(H(Z) \neq Y)}}$$

Proof. We first start by recalling Theorem 2 in Mozannar and Sontag (2020):

$$\begin{aligned}
L^{0-1}_{\text{def}}(\hat{m}^*, \hat{r}^*) \leq \hat{L}^{0-1}_{\text{def}}(\hat{m}^*, \hat{r}^*) &+ \mathfrak{R}_n(\mathcal{M}) + \mathfrak{R}_n(\mathcal{R}) + \mathfrak{R}_{n\mathbb{P}(H(Z) \neq Y)/2}(\mathcal{R}) \\
&+ 2 \sqrt{\frac{\log\left(\frac{2}{\delta}\right)}{2n}} + \frac{\mathbb{P}(H(Z) \neq Y)}{2} \exp\left(- \frac{n \mathbb{P}(H(Z) \neq Y)}{8} \right)
\end{aligned} \tag{30}$$

Note that here we avoid going through the optimal solution and just relate distribution performance to the training performance.

In the bound (30), $\mathfrak{R}_n(\mathcal{M})$ and $\mathfrak{R}_n(\mathcal{R})$ denote the Rademacher complexity of a halfspace in d dimensions where the infinity norm of each element in the halfspace is constrained by K_m and K_r respectively. Let us now compute this Rademacher complexity, inspired by Kakade and Tewari (2008):

$$
\begin{aligned}
\mathfrak{R}_n(\mathcal{M}) &= \frac{1}{n}\mathbb{E}\left[\sup_{M:||M||_\infty \leq K_m} \sum_{i=1}^{n} \epsilon_i M^\top x_i\right] \\
&\leq \frac{1}{n}\mathbb{E}\left[\sup_{M:||M||_1 \leq dK_m} M^\top \sum_{i=1}^{n} \epsilon_i x_i\right] \quad \text{(since } ||M||_1 \leq d||M||_\infty\text{)} \\
&= \frac{dK_m}{n}\mathbb{E}\left[\sum_{i=1}^{n} ||\epsilon_i x_i||_\infty\right] \\
&= \frac{dK_m}{n}\mathbb{E}\left[\sup_j \sum_{i=1}^{n} \epsilon_i [x_i]_j\right] \\
&\leq \frac{dK_m\sqrt{2\log d}}{n} \sup_j \sqrt{\sum_{i=1}^{n} [x_i]_j^2} \quad \text{(Massart's finite lemma on } x_{ij}\text{)} \\
&\leq \frac{dK_m\sqrt{2\log d}}{\sqrt{n}} \quad \text{(assume } ||x_i||_1 \leq 1 \text{ for all } i\text{)}
\end{aligned}
$$

Let us use the Rademacher complexity calculation in the bound to get:

$$
\begin{aligned}
L_{\text{def}}^{0-1}(\hat{m}^*, \hat{r}^*) \leq \hat{L}_{\text{def}}^{0-1}(\hat{m}^*, \hat{r}^*) + \frac{dK_m\sqrt{2\log d}}{\sqrt{n}} + \frac{dK_r\sqrt{2\log d}}{\sqrt{n}} + \frac{dK_m\sqrt{2\log d}}{\sqrt{n\mathbb{P}(H(Z) \neq Y)}} \\
+ 2\sqrt{\frac{\log\left(\frac{2}{\delta}\right)}{2n}} + \frac{\mathbb{P}(H(Z) \neq Y)}{2}\exp\left(-\frac{n\mathbb{P}(H(Z) \neq Y)}{8}\right)
\end{aligned}
$$

note that $\frac{\mathbb{P}(H(Z)\neq Y)}{2}\exp\left(-\frac{n\mathbb{P}(H(Z)\neq Y)}{8}\right)$ is a term that does not depend on the optimization and shrinks much faster than $\frac{8}{\sqrt{n\mathbb{P}(H(Z)\neq Y)}}$, so that we can summarize things as:

$$
L_{\text{def}}^{0-1}(\hat{m}^*, \hat{r}^*) \leq \hat{L}_{\text{def}}^{0-1}(\hat{m}^*, \hat{r}^*) + \frac{(K_m + K_r)d\sqrt{2\log d} + 10\sqrt{\log(2/\delta)}}{\sqrt{n\mathbb{P}(H(Z) \neq Y)}} \tag{31}
$$

□

D.4 Section 6 (`RealizableSurrogate`)

Theorem 2. *The* `RealizableSurrogate` L_{RS} *is a realizable* $(\mathcal{M}, \mathcal{R})$*-consistent surrogate for* L_{def}^{0-1} *for model classes closed under scaling, and satisfies* $L_{\text{def}}^{0-1}(m, r) \leq L_{RS}(m, r)$ *for all* (m, r)*.*

Proof. Let us recall the `RealizableSurrogate` loss pointwise:

$$
L_{RS}(\mathbf{g}, x, y, h) = -2\log\left(\frac{\exp(g_y(x)) + \mathbb{I}_{h=y}\exp(g_\perp(x))}{\sum_{y' \in \mathcal{Y}\cup\perp}\exp(g_{y'}(x))}\right) \tag{32}
$$

where $\mathbf{g} = \{g_i\}_{i\in\mathcal{Y}\cup\perp}$. Recall that the classifier and rejector are defined as: $m(x) = \arg\max_{y\in\mathcal{Y}} g_y(x)$ and $r(x) = \mathbb{I}_{\max_{y\in\mathcal{Y}} g_y(x) \leq g_\perp(x)}$.

We first prove that for every point, the `RealizableSurrogate` loss upper bounds the system 0-1 error: $L_{\text{def}}^{0-1}(m, r, x, y, h) \leq L_{RS}(\mathbf{g}, x, y, h)$:

1. **Case 1:** consider $r(x) = 0$ (classifier predicts):
   (a) **Case 1a:** if the classifier is incorrect, $\mathbb{I}_{m(x)\neq y} = 1$:

i. **Case 1ai:** If the human is incorrect, $\mathbb{I}_{h=y} = 0$:
then the loss is:$-2\log\left(\frac{\exp(g_y(x))}{\sum_{y'\in\mathcal{Y}\cup\perp}\exp(g_{y'}(x))}\right)$, we know since the classifier is incorrect, then it must be that $\frac{\exp(g_y(x))}{\sum_{y'\in\mathcal{Y}\cup\perp}\exp(g_{y'}(x))} \leq 0.5$ (since g_y is not the max), thus the loss is greater than 2 (log is base 2), and the 0-1 loss is 1 in this case.

ii. **Case 1aii**: if the human is correct then $\mathbb{I}_{h=y} = 1$:
then the loss is:$-2\log\left(\frac{\exp(g_y(x))+\exp(g_\perp(x))}{\sum_{y'\in\mathcal{Y}\cup\perp}\exp(g_{y'}(x))}\right)$, we know since the classifier is incorrect, then it must be that $\frac{\exp(g_y(x))}{\sum_{y'\in\mathcal{Y}\cup\perp}\exp(g_{y'}(x))} + \frac{\exp(g_\perp(x))}{\sum_{y'\in\mathcal{Y}\cup\perp}\exp(g_{y'}(x))} < 2/3$ since g_y is not the max neither is $g_\perp$, otherwise if the sum of these two fractions is greater than 2/3, then $\max_i \frac{\exp(g_i(x))}{\sum_{y'\in\mathcal{Y}\cup\perp}\exp(g_{y'}(x))} < 1/3$ then the maximum must be one of y or $\perp$ which is a contradiction. Finally, the loss is greater then $-2\log(2/3) = 1.17$ which is greater than 1.

(b) **Case 1b:** if the classifier is correct $\mathbb{I}_{m(x)=y} = 1$, then the 0-1 error is 0, since the `RealizableSurrogate` loss is ≥ 0 then it is an upper bound.

2. **Case 2:** consider $r(x) = 1$ (human predicts):

(a) **Case 2a:** if the human is correct then $\mathbb{I}_{h=y} = 1$:
then the 0-1 error is 0, since the `RealizableSurrogate` loss is ≥ 0 then it is an upper bound.

(b) if the human is incorrect then $\mathbb{I}_{h=y} = 0$:
the loss is $-2\log\left(\frac{\exp(g_y(x))}{\sum_{y'\in\mathcal{Y}\cup\perp}\exp(g_{y'}(x))}\right)$, we know since we defer, then it must be that $\frac{\exp(g_y(x))}{\sum_{y'\in\mathcal{Y}\cup\perp}\exp(g_{y'}(x))} \leq 0.5$ (since g_y is not the max), thus the loss is greater than 2 (log is base 2), and the 0-1 loss is 1 in this case.

this concludes the proof of the upper bound.

We now prove that L_{RS} is a realizable-consistent loss function.

Consider a data distribution and a human under which there exists $m^*, r^* \in \mathcal{M}\times\mathcal{R}$ that have zero error $L^{0-1}_{\text{def}}(m^*, r^*) = 0$. Associated with m^*, r^*, is a set of functions $\mathbf{g}^* \in \mathcal{G}$ that give rise to m^*, r^*. Let $\hat{\mathbf{g}}$ be the minimizer of the surrogate loss L_{RS} and the associated classifier and rejector be $\hat{m}, \hat{r}$.

We now upper bound the 0-1 loss of the pair $\hat{m}, \hat{r}$. Let $u \in \mathbb{R}$ be any real number:

$$
\begin{aligned}
&L^{0-1}_{\text{def}}(\hat{m}, \hat{r}) \\
&\leq L_{RS}(\hat{m}, \hat{r}) \quad \text{(loss is upper bound)} \\
&\leq L_{RS}(um^*, ur^*) \quad \text{(since } \hat{m}, \hat{r} \text{ is optimal for } L_{RS} \text{ and } \mathcal{M}\times\mathcal{R} \text{ is closed under scaling)} \\
&= \mathbb{E}[L_{RS}(um^*, ur^*, x, y, h)|r^* = 1]\mathbb{P}(r^* = 1) + \mathbb{E}[L_{RS}(um^*, ur^*, x, y, h)|r^* = 0]\mathbb{P}(r^* = 0) \qquad (33)
\end{aligned}
$$

Let us investigate the two terms in equation (33).

The first term is when $r^* = 1$, then we must have $g^*_\perp > \max_y g^*_y$ and $\mathbb{I}_{h=y} = 1$ since the data is realizable and when we defer the human must be correct. Examining the first term and taking the limit:

$$
\begin{aligned}
&\lim_{u\to\infty} \mathbb{E}[L_{RS}(um^*, ur^*, x, y, h)|r^* = 1]\mathbb{P}(r^* = 1) \\
&= \lim_{u\to\infty} \mathbb{E}[-2\log\left(\frac{\exp(ug^*_y(x)) + \mathbb{I}_{h=y}\exp(ug^*_\perp(x))}{\sum_{y'\in\mathcal{Y}\cup\perp}\exp(ug^*_{y'}(x))}\right)|r^* = 1]\mathbb{P}(r^* = 1) \\
&= \lim_{u\to\infty} \mathbb{E}[-2\log\left(\frac{\exp(ug^*_y(x)) + \exp(ug^*_\perp(x))}{\sum_{y'\in\mathcal{Y}\cup\perp}\exp(ug^*_{y'}(x))}\right)|r^* = 1]\mathbb{P}(r^* = 1) \\
&= \mathbb{E}[-2\log(1)\,|r^* = 1]\mathbb{P}(r^* = 1) = 0 \quad \text{(applying monotone convergence theorem)}
\end{aligned}
$$

The second term is when $r^* = 0$, then we must have $g^*_y > \max_{y'\in(\mathcal{Y}\setminus y)\cup\perp} g^*_{y'}$ since the data is realizable. Examining the second term and taking the limit:

$$\lim_{u\to\infty} \mathbb{E}[L_{RS}(um^*, ur^*, x, y, h)|r^* = 0]\mathbb{P}(r^* = 0)$$
$$= \lim_{u\to\infty} \mathbb{E}[-2\log\left(\frac{\exp(ug^*_y(x)) + \mathbb{I}_{h=y}\exp(ug^*_\perp(x))}{\sum_{y'\in\mathcal{Y}\cup\perp}\exp(ug^*_{y'}(x))}\right)|r^* = 0]\mathbb{P}(r^* = 0)$$
$$= \mathbb{E}[-2\log(1)\,|r^* = 0]\mathbb{P}(r^* = 0) = 0 \quad \text{(applying monotone convergence theorem)}$$

Thus combining the above two derivations, we obtain:

$$L^{0-1}_{\text{def}}(\hat{m}, \hat{r}) \leq 0.$$

We just proved that the optimal solution from minimizing `RealizableSurrogate` leads to a zero error solution in terms of system error which proves that the loss is realizable $(\mathcal{M}, \mathcal{R})$-consistent.

□

Theorem D.4. *The CrossEntropySurrogate L_{CE} (Mozannar and Sontag, 2020) is not a realizable $(\mathcal{M}, \mathcal{R})$-consistent surrogate for L^{0-1}_{def}.*

Proof. To prove that the surrogate L_{CE} is not realizable-consistent, we will construct an example with a data distribution and a model class closed under scaling such that: 1) there exists a zero error solution in the model class and 2) the minimizer of L_{CE} has non-zero error.

Consider the data distribution illustrated and described in Figure 10 consisting of four regions R0,R1,R2 and R3. Each region respectively has mass $1/4 + \alpha, 1/4, 1/4 - \alpha, 1/4$. Each region respectively has label $Y = 0, Y = 1, Y = 0, Y = 2$. The Human is perfectly accurate on Region 0 and inaccurate on every other region.

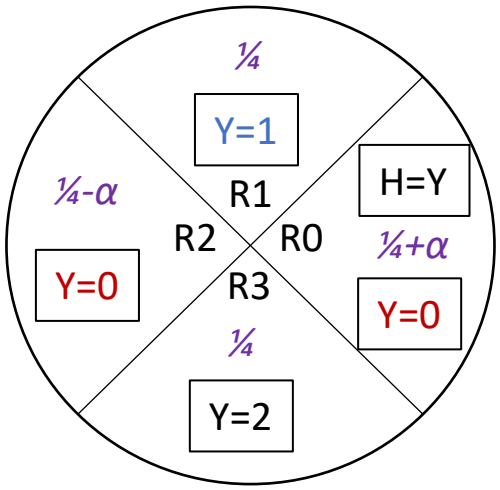


Figure 10: Data Distribution for our example: the data consists of four regions R0,R1,R2 and R3. Each region respectively has mass $1/4 + \alpha, 1/4, 1/4 - \alpha, 1/4$. Each region respectively has label $Y = 0, Y = 1, Y = 0, Y = 2$. The Human is only accurate on Region 0.

We consider a hypothesis class $\mathcal{F}$ parameterized by a scalar $c \in \mathbb{R}$ and four indices each in $i_0, i_1, i_2, i_\perp \in \{0, 1, 2, 3\}$. Let $f_i(x) = c\mathbb{I}\{x \in R_i\}$, a function $f \in \mathcal{F}$ defines a rejector and classifier as: $m(x) = \arg\max\{c \cdot f_{i0}(x), c \cdot f_{i1}(x), c \cdot f_{i2}(x)\}$ (ties are decided uniformly randomly) and $r(x) = \mathbb{I}\{c \cdot f_\perp(x) > \max\{c \cdot f_{i0}(x), c \cdot f_{i1}(x), c \cdot f_{i2}(x)\}$. This hypothesis class is closed under scaling.

The error minimizing function f^* in this hypothesis class is obtained by setting $c > 0, i_0 = 2, i_1 = 1, i_2 = 3, i_\perp = 0$ which obtains zero 0-1 error. No solution with $c < 0$ is optimal, since the maximum will always coincide with at least two labels and we break ties in a consistent fashion. This data distribution and hypothesis class is realizable.

Surrogate solution. We will argue that one can obtain a lower L_{CE} loss by deviating from the optimal solution f^*. The intuition for why this is the case is that the L_{CE} penalizes misclassifying points even when they are deferred. Hence, when

α is sufficiently large, L_{CE} will try to classify the more probable region R0 as label 0 instead of simply deferring on this region and classifying region R2 as label 0.

Consider the function $\hat{f}$ defined with arbitrary $c > 0$ and $i_0 = 0, i_1 = 1, i_2 = 3, i_\perp = 0$—note that this function disagrees with the optimal solution on i_0 only. Fixing c, we will compute the difference of L_{CE} loss between $\hat{f}$ and f^* with the same c, this defines only a deviation in terms of i_0. We will compute the difference in each region separately.

Region 1 and Region 3: On both region 1 and region 3, the difference will be shown to be zero. In both regions, the human is incorrect and note that i_1 and i_2 are identical in both solutions. The loss of $\hat{f}$ in region 1 is:

$$-\frac{1}{4}\log\left(\frac{e^c}{3+e^c}\right)$$

this is the same as the loss of f^*, by symmetry the loss is the same in region 3.

We will now compute the sum of the difference in region 2 and region 0:

Region 2: In this region the human is also incorrect, the difference in the loss of $\hat{f}$ and f^* is:

$$\mathbb{E}_{x\in R2}[L_{CE}(f^*) - L_{CE}(\hat{f})] = (\frac{1}{4} - \alpha)\cdot\left(\log\left(\frac{1}{4}\right) - \log\left(\frac{e^c}{3+e^c}\right)\right) \in [-(\frac{1}{4}-\alpha)\log(4), 0]$$

Region 0: In this region the human is correct, the difference is :

$$\begin{aligned}&\mathbb{E}_{x\in R0}[L_{CE}(f^*) - L_{CE}(\hat{f})]\\ &= (\frac{1}{4}+\alpha)\cdot\left(-\log\left(\frac{1}{3+e^c}\right) - \log\left(\frac{e^c}{3+e^c}\right) + \log\left(\frac{e^c}{2+2e^c}\right) + \log\left(\frac{e^c}{2+2e^c}\right)\right)\end{aligned}$$

To compute the difference in the loss between $\hat{f}$ and f^*, we sum the difference in Region 2 and Region 0:

$$\begin{aligned}&L_{CE}(f^*) - L_{CE}(\hat{f})\\ &= \frac{1}{4}\left(\log\left(\frac{1}{4}\right) - \log\left(\frac{e^c}{3+e^c}\right) - \log\left(\frac{1}{3+e^c}\right) - \log\left(\frac{e^c}{3+e^c}\right) + 2\log\left(\frac{e^c}{2+2e^c}\right)\right)\\ &+ \alpha\left(-\log\left(\frac{1}{3+e^c}\right) + 2\log\left(\frac{e^c}{2+2e^c}\right) - \log\left(\frac{1}{4}\right)\right)\\ &= -(\frac{1}{4}+\alpha)\left(\log\left(\frac{1}{3+e^c}\right) - 2\log\left(\frac{e^c}{2+2e^c}\right)\right) - \frac{1}{2}\log\left(\frac{e^c}{3+e^c}\right) + (\frac{1}{4}-\alpha)\log\left(\frac{1}{4}\right)\end{aligned}$$

We can simplify this difference to further become:

$$\frac{1}{4}\left(8\alpha c - 2\log(4) - 2(1+4\alpha)\log(1+e^c) + (3+4\alpha)\log(3+e^c)\right)$$

Note that when $c = 0$, the above difference is 0. Let us set $\alpha = 0.125$ for concreteness (other values of α also work, in particular larger values, but not all smaller values). We compute the derivative of the difference with respect to c, obtaining:

$$\begin{aligned}\frac{d}{dc}(L_{CE}(f^*) - L_{CE}(\hat{f})) &= \frac{1}{4}\left(\frac{3.5e^c}{e^c+3} - \frac{3e^c}{e^c+1} + 1\right)\\ &= \frac{0.375(2 - e^c + e^{2c}))}{(1+e^c)(3+e^c)} > 0\end{aligned}$$

We just showed that the difference has derivative strictly larger than 0 with respect to c, moreover the difference is 0 when $c = 0$, thus when $c > 0$ the difference is strictly bigger than 0.

We just proved that with respect to the surrogate loss L_{CE}, the optimal solution with respect to L_{def}^{0-1} is not optimal, thus the surrogate is not a realizable $(\mathcal{M}, \mathcal{R})$-consistent surrogate for L_{def}^{0-1}

□