Optimizing AI for Teamwork

Gagan Bansal[1] , **Besmira Nushi**[2] , **Ece Kamar**[2] , **Eric Horvitz**[2] , **Daniel S. Weld**[1,3]
[1]University of Washington
[2]Microsoft Research
[3]Allen Institute for Artificial Intelligence

Abstract

In many high-stakes domains such as criminal justice, finance, and healthcare, AI systems may recommend actions to a human expert responsible for final decisions, a context known as *AI-advised decision making*. When AI practitioners deploy the most *accurate* system in these domains, they implicitly assume that the system will function alone in the world. We argue that the most accurate AI team-mate is not necessarily the *best teammate*; for example, predictable performance is worth a slight sacrifice in AI accuracy. So, we propose training AI systems in a human-centered manner and directly optimizing for *team performance*. We study this proposal for a specific type of human-AI team, where the human overseer chooses to *accept* the AI recommendation or *solve* the task themselves. To optimize the team performance we maximize the team's *expected utility*, expressed in terms of quality of the final decision, cost of verifying, and individual accuracies. Our experiments with linear and non-linear models on real-world, high-stakes datasets show that the improvements in utility while being small and varying across datasets and parameters (such as cost of mistake), are real and consistent with our definition of team utility. We discuss the shortcoming of current optimization approaches beyond well-studied loss functions such as *log-loss*, and encourage future work on human-centered optimization problems motivated by human-AI collaborations.

1 Introduction

Increasingly, humans work collaboratively with an AI teammate, for example, because the team may perform better than either the AI or human alone [Nagar and Malone, 2011; Patel *et al.*, 2019; Kamar *et al.*, 2012], or because legal requirements may prohibit complete automation [GDPR, 2020; Nickelsburg, 2020]. For human-AI teams, just like for any team, optimizing the performance of the whole team is more important than optimizing the performance of an individual member. Yet, to date for the most part, the AI community has focused on maximizing the individual accuracy of machine-learning models. This raises an important question: Is the most accurate AI the best possible teammate for a human?

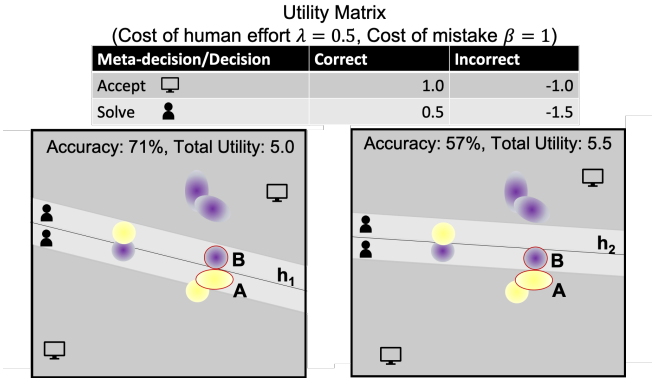


Figure 1: In a human-AI team, a more accurate classifier (h_1, left pane, learned using log-loss) may produce lower *team utility* than a less accurate one (h_2, right pane). Suppose the human can either quickly *accept* the AI's recommendation or *solve* the task themselves, incurring a cost λ, to yield a more reliable result. The payoff matrix describes the utility of different outcomes. One optimal policy is for the human to accept recommendations when the AI is confident, but verify uncertain predictions (shown in the light grey region surrounding each hyperplane). While h_2 is less accurate than h_1 (because B is incorrectly classified), it results in a higher team utility: Since h_2 moved A outside the verify region, there are more *correctly classified* inputs on which the user can rely on the system.

We argue that the answer is "No." We show this formally, but the intuition is simple. Consider human-human teams, *Is the best-ranked tennis player necessarily the best doubles teammate?* Clearly not — teamwork puts additional demands on participants besides high individual performance, such as ability to complement and coordinate with one's partner. Similarly, creating high-performing human-AI teams may require training AI that exhibits additional *human-centered* properties that facilitate trust and delegation. Implicitly, this is the motivation behind much work in intelligible AI [Caruana *et al.*, 2015; Weld and Bansal, 2019] and post-hoc explainable AI [Ribeiro *et al.*, 2016], but we suggest that directly modeling the collaborative process may offer additional benefits.

Recent work emphasized the importance of better under-

standing how people *transform AI recommendations into decisions* [Kleinberg *et al.*, 2018]. For instance, consider scenarios when a system outputs a recommendation on which it is uncertain. A rational user is likely to distrust such recommendations — erroneous recommendations are often correlated with a low confidence in prediction [Hendrycks and Gimpel, 2018]. In this work we assume that the user will discard the recommendation and *solve* the task themselves, after incurring a cost (*e.g.*, due to additional human effort). As a result, the team performance depends on the AI accuracy only in the *accept region*, *i.e.*, the region where a user is actually likely to rely on AI t the singular objective of optimizing for AI accuracy (*e.g.*, using *log-loss*) may hurt team performance when the model has fixed inductive bias; team performance will instead benefit from improving AI in the accept regions in Figure 1. While there exist other aspects of collaboration that can also be addressed via optimization techniques, such as model interpretability, supporting complementary skills, or enabling learning among partners, the problem we address in this paper is to account for team-based utility as a basis for collaboration.

In sum, we make the following contributions:

1. We highlight a novel, important problem in the field of human-centered artificial intelligence: the most *accurate* ML model may not lead to the highest *team utility* when paired with a human overseer.
2. We show that *log-loss*, the most popular loss function, is insufficient (as it ignores *team utility*) and develop a new loss function *team-loss*, which overcomes its issues by calculating a team's expected utility.
3. We present experiments on multiple real-world datasets that compare the gains in utility achieved by *team-loss* and *log-loss*. We observed that while the gains are small and vary across datasets they reflect the behavior encoded in the loss. We present further analysis to understand how *team-loss* results in a higher utility and when, for example, as a function of domain parameters such as cost of mistake.

2 Problem Description

We focus on a special case of AI-advised decision making where a classifier h gives recommendations to a human decision maker to help make decisions (Figure 2a). If $h(x)$ denotes the classifier's output, a probability distribution over $\mathcal{Y}$, the recommendation r consists of a label $\hat{y} = \arg\max h(x)$ and a confidence value $\max h(x)$, *i.e.*, $r := (\hat{y}, \max h(x))$. Using this recommendation, the user computes a final decision d. The environment, in response, returns a utility which depends on the quality of the final decision and any cost incurred due to human effort. Let U denote the utility function. If the team classifies a sequence of instances, the objective of this team is to maximize the cumulative utility. Before deriving a closed form equation of the objective, we characterize the form of the human-AI collaboration along with our assumptions. We study this particular, simple setting as a first step to explore the opportunities and challenges in team-centric optimization. If we cannot optimize for this simple setting, it may be much harder to optimize for more complex scenarios (discussed more in Section 4).

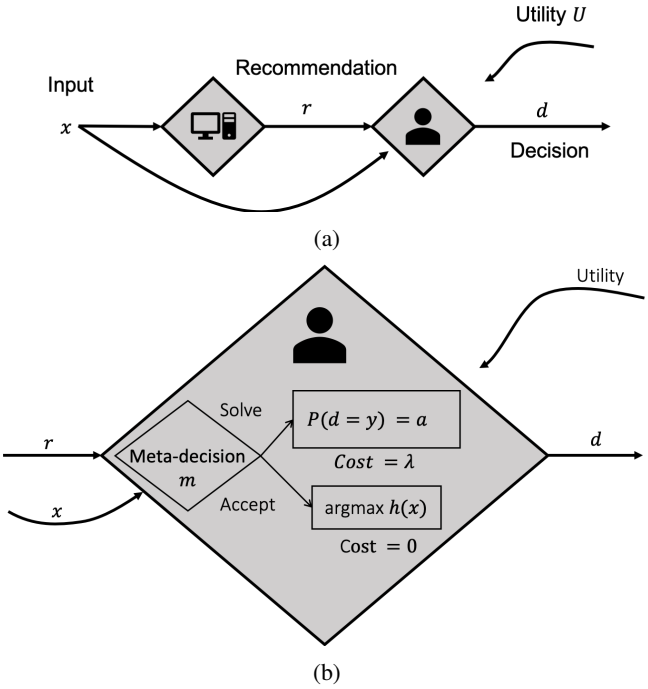


Figure 2: (a) A schematic of AI-advised decision making. (b) To make a decision, the human decision maker either accepts or overrides a recommendation. The `Solve` meta-decision is costlier than `Accept`.

1. *User either accepts the recommendation or solves the task themselves:* The human computes the final decision by first making a meta-decision: `Accept` or `Solve` (Figure 2b). `Accept` passes off the recommendation as the final decision. In contrast, `Solve` ignores the recommendation and the user computes the final decision themselves. Let m denote the function that maps an input instance and recommendation to a meta-decision in $\mathcal{M} = \{\texttt{Accept}, \texttt{Solve}\}$. As a result, the optimal classifier h^* would maximize the team's expected utility:

$$h^* = \arg\max_h \mathbb{E}_{x,y}[U(m, d)] \tag{1}$$

2. *Mistakes are costly:* A correct decision results in unit reward whereas an incorrect decision results in a penalty $\beta \geq 1$.
3. *Solving the task is costly:* Since it takes time and effort for the human to perform the task themselves, (*e.g.*, cognitive effort), we assume that the `Solve` meta-decision costs more than `Accept`. Further, without loss of generality, we assume λ units of cost to `Solve` and zero cost to `Accept`.

   Using the above assumptions we obtain the following utility function. The values in each cell of the table originate from subtracting the cost of the action from the environment reward.

Symbol	Description
a	Human accuracy
$\beta \in \mathbb{R}^+$	Cost of mistake
$h(x)[\hat{y}] = \max h(x)$	Confidence in the predicted label
$d : \mathcal{X} \times \mathcal{R} \to \mathcal{Y}$	Human decision maker
$h : \mathcal{X} \to [0,1]^{\lvert\mathcal{Y}\rvert}$	Classifier
$\mathcal{H}$	Classifier hypothesis space
$\lambda \in \mathbb{R}^+$	Cost of human effort to `Solve`
$m : \mathcal{X} \times \mathcal{R} \to \mathcal{M}$	Meta-decision function
$\mathcal{M} \coloneqq \{\texttt{Accept}, \texttt{Solve}\}$	Meta-decision space
$\psi : \mathcal{H} \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$	Expected team utility
$r \in \mathcal{R}$	Recommendation
$\mathcal{R} \coloneqq \mathcal{Y} \times [0,1]$	Recommendation space
$U : \mathcal{M} \times \mathcal{Y} \to \mathbb{R}$	Utility function
$\mathcal{X}$	Feature space
$\hat{y} \in \mathcal{Y}$	Recommended label
$\mathcal{Y}$	Label space

Table 1: Notation.

Meta-decision/Decision	Correct	Incorrect
`Accept` [A]	1	$-\beta$
`Solve` [S]	$1-\lambda$	$-\beta-\lambda$

Figure 3: Team utility w.r.t. meta-decision and decision accuracy.

4. *Human is uniformly accurate:* Let $a \in [0,1]$ denote the conditional probability that if the user solves the task, they will make the correct decision, *i.e.*,

$$P(d = y | m = \texttt{S}) = a \tag{2}$$

5. *Human is rational:* The user makes the meta-decision by comparing expected utilities. Further, the user trusts the classifier's confidence as an accurate indicator of the recommendation's reliability. As a result, the user will choose `Accept` if and only if the expected utility for accepting is higher than the expected utility for solving.

$$\begin{aligned} \mathbb{E}[U(m = \texttt{A})] &\geq \mathbb{E}[U(m = \texttt{S})] \\ h(x)[\hat{y}] - (1 - h(x)[\hat{y}]) \cdot \beta &\geq a - (1-a) \cdot \beta - \lambda \\ h(x)[\hat{y}] &\geq a - \frac{\lambda}{1+\beta} \end{aligned}$$

Let $c(\beta, \lambda, a)$ denote the minimum value of system confidence for which the user's meta-decision is `Accept`.

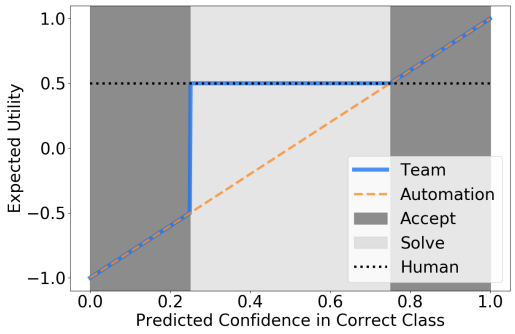
$$c(\beta, \lambda, a) = a - \frac{\lambda}{1+\beta} \tag{3}$$

This implies the human will follow the following threshold-based policy to make meta-decisions:

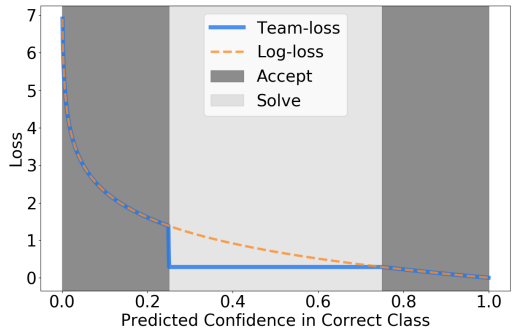
$$P(m = \texttt{A}) = \begin{cases} 1 & \text{if } h(x)[\hat{y}] \geq c(\beta, \lambda, a) \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

2.1 Expected Team Utility

We now derive the equation for expected utility of recommendations. Let $\psi(h)$ denote the expected utility of the classifier h and a decision maker d.



(a) In the `Accept` region the expected team utility is equal to expected automation utility, while in the `Solve` region it is the same as the human utility. The negative team utility in the left-most region indicates scenarios where AI gives high-confidence but incorrect recommendations to the human.



(b) In the `Accept` region, *team-loss* behaves similar to *log-loss*, however in the `Solve` region it results in a constant loss.

Figure 4: Visualization of expected utility and loss. This visualization corresponds to the case when $\lambda = 0.5, \beta = 1$, and $a = 1$ (*i.e.*, the human is perfectly accurate but it costs them half a unit of utility to solve the task).

$$\psi(x, y) = \mathbb{E}[U(m, d)]$$

$$\begin{aligned} \psi(x, y) = P(m = \texttt{A}) \cdot \Big[& P(d = y | m = \texttt{A}) \cdot 1 \\ & + P(d \neq y | m = \texttt{A}) \cdot (-\beta)) \Big] \\ & + P(m = \texttt{S}) \cdot \Big[P(d = y | m = \texttt{S}) \cdot (1 - \lambda) \\ & + P(d \neq y | m = \texttt{S}) \cdot (-\beta - \lambda) \Big] \end{aligned}$$

Since upon `Accept`, the human returns the classifier's recommendation, the probability that the final decision is correct is the same as the classifier's predicted probability of the correct decision:

$$P(d = y | m = \texttt{A}) = h(x)[y] \tag{5}$$

Using Equations 2 and 5, we obtain the following equation for expected utility of team.

$$\begin{aligned} &= P(m = \texttt{A}) \cdot \Big[(1+\beta)\cdot h(x)[y] - \beta\Big] \\ &+ P(m = \texttt{S}) \cdot \Big[(1+\beta)\cdot a - \beta - \lambda\Big] \\ &= P(m = \texttt{A}) \cdot \Big[(1+\beta)\cdot (h(x)[y] - a) + \lambda\Big] \\ &+ \underbrace{\Big[(1+\beta)\cdot a - \beta - \lambda\Big]}_{\text{constant}} \end{aligned} \tag{6}$$

Using Equations 4 and 6, we obtain the following expression for expected utility:

$$\psi(x, y) = \begin{cases} (1+\beta)\cdot h(x)[y] - \beta & \text{if } h(x)[\hat{y}] \geq c(\beta, \lambda, a) \\ (1+\beta)\cdot a - \beta - \lambda & \text{otherwise} \end{cases} \tag{7}$$

Figure 4a visualizes the expected team utility of the classifier predictions as a function of confidence in the true label.

2.2 Utility-Based Loss

Since gradient descent-based minimization of loss functions is common in machine learning, we transform the expected utility ψ into a loss function by negating it. We call this new loss function *team-loss*.

$$\mathcal{L}_{team}(x, y) = -\log(\psi(x, y)) \tag{8}$$

We take a logarithm before negating utility to allow comparisons with *log-loss*, where the logarithmic nature of loss is known to benefit optimization, for example, by heavily penalizing high-confidence mistakes.[1] Figure 4b visualizes this new loss function.

3 Experiments

We conducted experiments to answer the following research questions:

RQ1 Does the new loss function result in a classifier that improves team utility over the most accurate classifier?

RQ2 How do these improvements change with properties of the task, *e.g.*, the cost of mistakes (β)?

RQ3 How do these improvements change with properties of the dataset, *e.g.*, with data distribution or dimensionality?

Metrics and Datasets We compared the utility achieved by two models: the most accurate classifier trained using *log-loss* and a classifier optimized using *team-loss* on the datasets described in Table 2. We experimented with two synthetic datasets and four real-world datasets with high-stakes. The real datasets are from domains that are known to or already deploy AI to assist human decision makers. The Scenario1 dataset refers to a dataset we created by sampling 10000 points from the data distribution described in Figure 1.

Dataset	#Features	#Examples	% Positive
Scenario1	2	10000	0.43
Moons	2	10000	0.50
German	24	1000	0.30
Fico	39	9861	0.52
Recidivism	13	6172	0.46
Mimic	714	21139	0.13

Table 2: We used two synthetic datasets (Scenario1, Moons) and four real-world datasets from high-stakes domains that are known to be used in AI-assisted decision making settings. The Mimic dataset has the most class imbalance.

Training Procedure We experimented with two models: logistic regression and multi-layered perceptron (two hidden layers with 50 and 100 units). For each task (defined by a choice of task parameters, dataset, model, loss) we optimized the loss using stochastic gradient descent (SGD) and also used standard, well-known training practices such as using regularization, check-pointing, and learning rate schedulers. We selected the best hyper-parameters using 5-fold cross validation, including values for the learning rate, batch size, patience and decay factor of the learning rate scheduler, and weight of the L2 regularizer.

In our initial experiments for training with *team-loss* using SGD, we observed that the classifier's loss would never reduce and in fact remain constant. This happened because, in practice, random initializations resulted in classifiers that are uncertain on all examples. And, since, by definition, *team-loss* is flat in these uncertain regions (Figure 4b), the gradients was zero and uninformative. To overcome this issue, we initialized the classifiers with the (already converged) most accurate classifier.

3.1 Results

RQ1: Experiments showed that when we used *team-loss*, the magnitude of improvements in team utility over *log-loss* varied across the datasets but were consistently observed (Table 3). We observed that *team-loss* often sacrifices classifier accuracy to improve team utility, the more desirable metric. For the linear classifier, this sacrifice is especially large on the synthetic datasets: Scenario1 (16%) and Moons (1%) datasets. For the MLP, *team-loss* sacrifices 2% accuracy to improve team utility.[2]

While the metrics in Table 3 (change in accuracy and utility) provide a global understanding of the effect of *team-loss*, they do not help understand *how team-loss* achieved improvements and whether the behavior of the new models is consistent with intuition. Figure 5 visualizes the difference in behavior (averaged over 150 seeds) between the classifiers produced by *log-loss* and *team-loss* on the Scenario1 dataset.

[1] Since loss can be negative, in our implementation, before computing a logarithmic of utility we appropriately shift up the utility function (by subtracting its minimum value).

[2] We report absolute improvements instead of percentage improvements in utility because utility can be negative.

Model	Dataset	Acc LL	Util LL	Δ Acc	Δ Util
Linear	Scenario1	0.86	0.59	-0.16	0.165
	Moons	0.89	0.81	-0.01	0.020
	German	0.75	0.61	-0.004	0.009
	Mimic	0.88	0.80	-0.000	0.001
	Recid	0.68	0.53	0.000	0.000
	Fico	0.73	0.58	0.000	-0.000
MLP	Fico	0.72	0.56	0.01	0.018
	Scenario1	0.98	0.84	-0.04	0.008
	Moons	1.00	0.99	0.00	0.007
	German	0.74	0.61	-0.02	0.003
	Mimic	0.88	0.80	0.00	0.003
	Recid	0.67	0.52	-0.00	0.001

Note: LL indicates *log-loss*

Table 3: Differences in performance (accuracy and utility) of *team-loss* and *log-loss* for all datasets (averaged over 150 runs). Datasets are sorted in descending order of improvements in utility and the analysis is divided by classifier type, linear and multi-layered perceptron. We observe that *team-loss* often sacrifices accuracy to improve utility. While the gains in utility are small they are consistently observed across datasets.

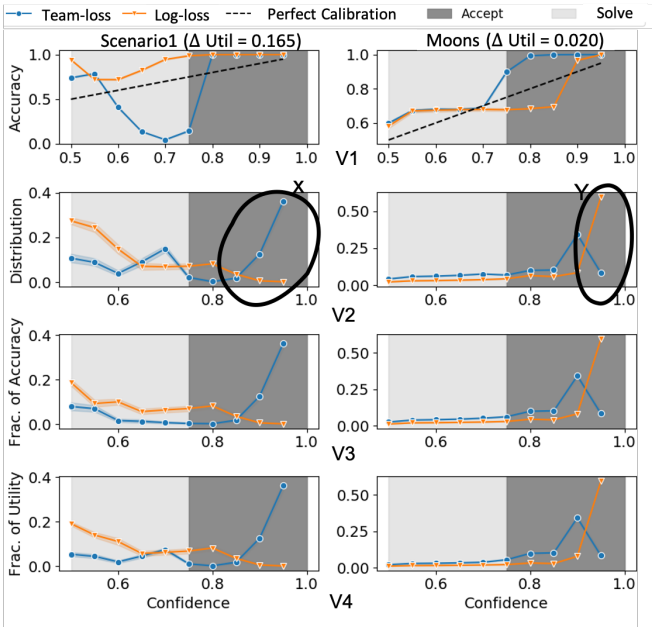


Figure 5: Differences between behavior of linear classifiers learned using *log-loss* and *team-loss* on the Scenario1 and Moons datasets (averaged over 150 runs). **Scenario1:** 1) *team-loss* sacrifices accuracy in the `Solve` region, 2) makes fewer predictions in the `Solve` region and more high-confidence predictions in the right-half of the `Accept` region (annotated as X), 3) reduces the contribution to system accuracy from `Solve` and increases it from the `Accept` region, 4) results in higher area under the curve indicating an increase in overall utility. **Moons:** 1) *team-loss* improves accuracy in the `Accept` region, 2) makes fewer very-high confidence predictions (marked as Y) and more moderately-high confidence predictions in the `Accept` region. Figure 6 shows similar visualizations for the real-world datasets.

Specifically, as shown in Figure 5, we visualize and compare their:

V1. Calibration using *reliability curves*, which compare system confidence and its true accuracy. A perfectly calibrated system, for example, will be 80% accurate on regions that is 80% confident. However, in practice, systems may over- or under-confident.

V2. Distributions of confidence in predictions. For example, in Figure 5, *team-loss* makes more high-confidence predictions than *log-loss*.

V3. Fraction of total system accuracy contributed by different regions (of confidence values). Thus, the area under this curve indicates the system's total accuracy. Note that for our setup the area under the curve in the `Accept` region is more crucial than the area in the `Solve` region since in the latter the human is expected to take over.

V4. Similar to (V4), the forth sub-graph shows the fraction of total system utility contributed by different regions of confidence.

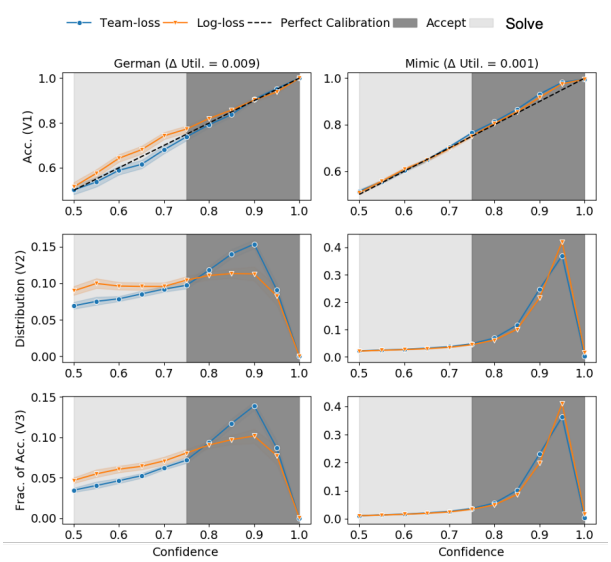
If *team-loss* had not resulted in different predictions than *log-loss*, the curves in Figure 5 for the two loss functions would have been indistinguishable. However, we observed that *team-loss* results in dramatically different predictions than *log-loss*. In fact, we noticed two types of behaviors when *team-loss* improved utility.

B1 The first type of behavior was observed on Scenario1 datatset (Figure 5) and is easier to understand as it matches the intuition we set out in the beginning– the classifier trained with *team-loss* sacrifices accuracy on the uncertain examples in the `Solve` region to make more high-confidence predictions in the `Accept` region. This change improves system accuracy in the `Accept` region, which is where the system accuracy matters and contributes to team utility. Later, we show that this same behavior is observed on the German dataset (Figure 6)

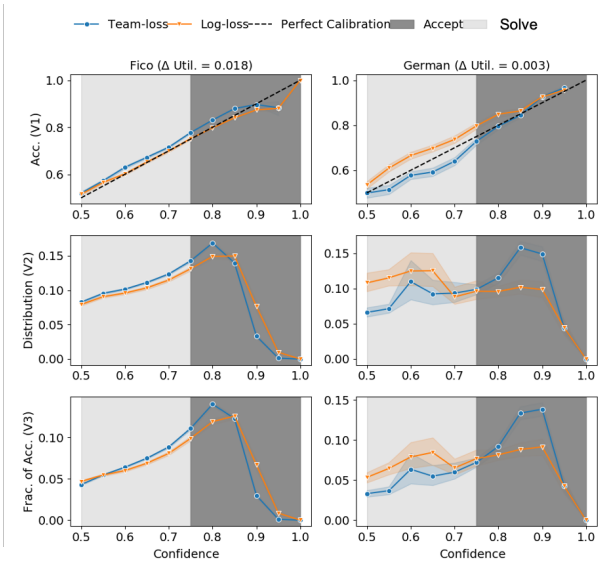
B2 The second type of behavior was observed on Moons (Figure 5), where the new loss *increases* accuracy in the `Accept` region at the cost predicting fewer very high-confidence predictions (*e.g.*, when confidence is greater than 0.95 in the region marked Y). This change improves utility because the system's accuracy in the `Accept` region matters more than making very high-confidence predictions.

In both these behaviors, *team-loss* effectively increases the contribution to AI accuracy from the `Accept` region, *i.e.*, the region where AI's performance providing value to the team. In contrast, *log-loss* has no such considerations. Figure 6 shows a similar analysis on the real datasets for both the linear and MLP classifiers. When *team-loss* improves utility, we see one of the two behaviors we described above.

RQ2: Since the penalty of mistakes may be task-dependant (*e.g.*, an incorrect diagnosis may be costlier than incorrect loan approval), we varied the mistake penalty β to study its effects on the improvements from *team-loss*. Our experiments (Figure 7) showed that the difference in utilities depend on the cost of mistake, and highest difference is



(a) **Linear classifier**: On German, we observed B1, where *team-loss* compared to *log-loss* preserved accuracy and made more predictions in the `Accept` region, and sacrificed accuracy and mass of prediction distribution in the `Solve` region. In contrast, on Mimic, we observed B2, where *team-loss* increased accuracy in the `Accept` region but made fewer very high confidence predictions (*e.g.*, confidence > 0.9).



(b) **MLP classifier:** On Fico, we observed a behavior B2 similar to Moons (Figure 5), where using *team-loss* increased the accuracy in the `Accept` region and reduced the number of very-high confidence predictions (same as moons for linear). In contrast on the German dataset, we observed a behavior B1 similar to the Scenario1 dataset (Figure 5), where using *team-loss* sacrificed accuracy in the `Solve` region and increased the number of predictions in the `Accept` region.

Figure 6: Comparison of the predictions of *log-loss* and *team-loss* on the real-world datasets when *team-loss* improves utility (150 seeds).

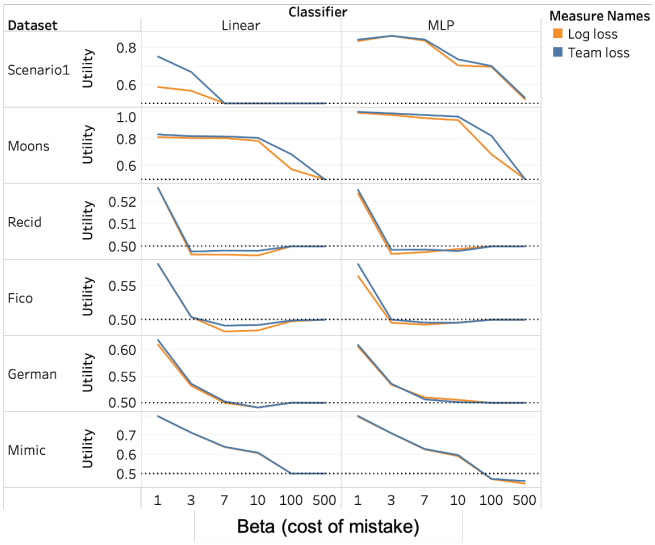


Figure 7: Comparison of utility achieved by the two loss functions. Across values of β, in most datasets, *team-loss* achieves higher utility than *log-loss*; however, the β value that results in the highest relative improvements is different across datasets. Interestingly, when *log-loss* results in lower utility than a human-only baseline (indicated by dotted line), *e.g.*, as seen on Recidivism as penalty increases, *team-loss* still attempts to nudge its utility to match the human baseline.

observed for a different value of β across datasets. We also observed that, for our setup, as the mistake penalty increases, *log-loss* may achieve lower performance than the human-only baseline, and so, deploying automation is undesirable in these cases. For example, on Fico and $\beta = 7$), linear model learned using *log-loss* achieves lower performance than human baseline. Similarly, on Mimic and $\beta = 500$, MLP learned using *log-loss* deploying the AI is undesirable.

Model	Dataset	Acc LL	Util LL	Δ Acc	Δ Util
Linear	German-b	0.72	0.56	-0.01	0.004
	Mimic-b	0.77	0.65	0.00	**0.002**
MLP	German-b	0.74	0.57	0.00	**0.024**
	Mimic-b	0.93	0.87	0.00	0.002

Table 4: Performance on German and Mimic datasets after correcting class imbalance. Bold indicates setting where balancing the dataset improved the gains in utility compared to its original version. We observed that for MLP, after balancing the German dataset, the gains in utility improved substantially, from 0.003 (see Table 3) to 0.024.

RQ3: Since the gains from using *team-loss* were small and varied across datasets, we conducted experiments to investigate properties of the dataset that may have affected these improvements. While there are many properties of a dataset one could investigate, we studied following:

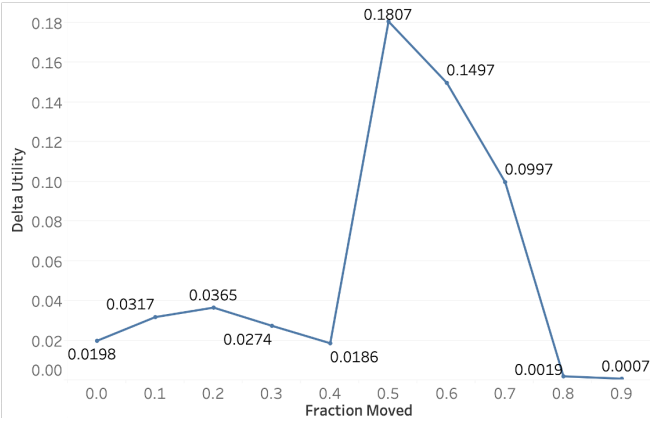


Figure 8: Relative performance of *team-loss* on the Moons dataset for linear classifier as we varied the data distribution and moved more points towards the edges. "Fraction Moved" indicates the fraction of total number of points that were moved towards the overlapping edges of the two moons.

1. *Data distribution* In the Moons dataset, we observed that the linear model trained with *team-loss* increased utility by increasing confidence on the examples on the outer edges of the moons enough to move these examples to the `Accept` region. So, to test whether a different data distribution would benefits from using *team-loss*, we created additional versions of Moons by systematically moving points from the middle of the circle towards its edges. Figure 9 shows the improvements in utility as we moved more data.
2. *Class imbalance* While most of our datasets were balanced, German and Mimic had a lower percentage of positive instances (see Table 2). We conducted experiments on balanced versions of these two datasets to understand if class imbalance affected our observations in the previous experiments. Table 4 shows the performance after we over-sampled the positive class to adjust for class imbalance in the two datasets. We observed that in both cases correcting class imbalance increased the improvement when using *team-loss*.

We also focussed on the dimensionality of the datasets. Since *team-loss* may be harder to optimize than NLL, an increase in data dimensions may affect the optimizer's ability (in this case SGD) to optimize *team-loss* objective. We also experimented with ADAM as the optimizer, unfortunately it did not provide any benefits. For a given dataset, we varied its dimensionality by using only a subset of features. However, we did not notice any correlations between dimensionality and improvements in utility.

4 Discussion

We conjecture two reasons to explain the small gains in utility on real datasets using *team-loss*: either there is no scope for improving the utility on those datasets and model pairs or our current optimization procedures are ineffective. Since we do not know the optimal (utility) solution for a given dataset and model, we cannot verify or reject the first conjecture. However, the results on the two synthetic datasets suggest the existence of situations where there is a significant gap between the utilities achieved using *team-loss* and *log-loss*.

However, it is possible that our current optimization procedures may be ineffective for optimizing *team-loss*. One reason this might happen is that *team-loss* is more complex that *log-loss*– it introduces new plateaus in the loss surface and thus may increase the chances of optimization methods such stochastic gradient descent getting stuck in local minima. In fact, in our experiments, we observed that on the datasets where *team-loss* did not increase utility it resulted in predictions identical to *log-loss*. This may, for example, happen if the most accurate classifier is a local minima. Since we use the most accurate classifier to initialize the optimization on *team-loss*, this entails that the further optimization with the new loss did not manage to overcome the potential local minima.

While we propose a solution for simplified human-AI teamwork (see assumptions in Section 2), our observations have implications for human-AI teams in general. If we cannot optimize utility for our simplified case, it may be harder to optimize utility in scenarios where users make `Accept` and `Solve` decisions using a richer, more complex mental model instead beyond relying on just model confidence. Such scenarios are common in cases where the system confidence is an unreliable indicator of performance (*e.g.*, due to poor calibration), and, as a result, the user develops an understanding of system failures in terms of domain features. For example, Tesla drivers often override the Autopilot using features such as road and weather conditions. We can reduce this case, where users have a complex mental model, to the one we studied. Specifically, we can construct a loss function that is constant when a prediction belongs to the `Solve` region described by the user's mental model and *log-loss* otherwise. This case may be harder to optimize because the resultant loss surface will contain more complex combinations of plateaus and local optima than the one we considered.

5 Related Work

Our approach is closely related to *maximum-margin classifiers*, such as an SVM optimized with the hinge loss [Burges, 1998], where a larger soft margin can be used to make high-confidence and accurate predictions. However, unlike our approach, it is not possible to directly plug the domain's payoff matrix (*e.g.*, in Figure 3) into such a model. Furthermore, the SVM's output and margin do not have an immediate probabilistic interpretation, which is crucial for our problem setting. One possible (though computationally intensive) solution direction is to convert margin into probabilities, *e.g.*, using post-hoc calibration (*e.g.*, Platt scaling [Platt, 1999]), and use cross-validation for selecting margin parameters to optimize team utility. While it is still an open question whether such an approach would be effective for SVM classifiers, in this work we focused our attention on gradient-based optimization.

Another related problem is *cost-sensitive learning*, where different mistakes incur different penalties; for example,

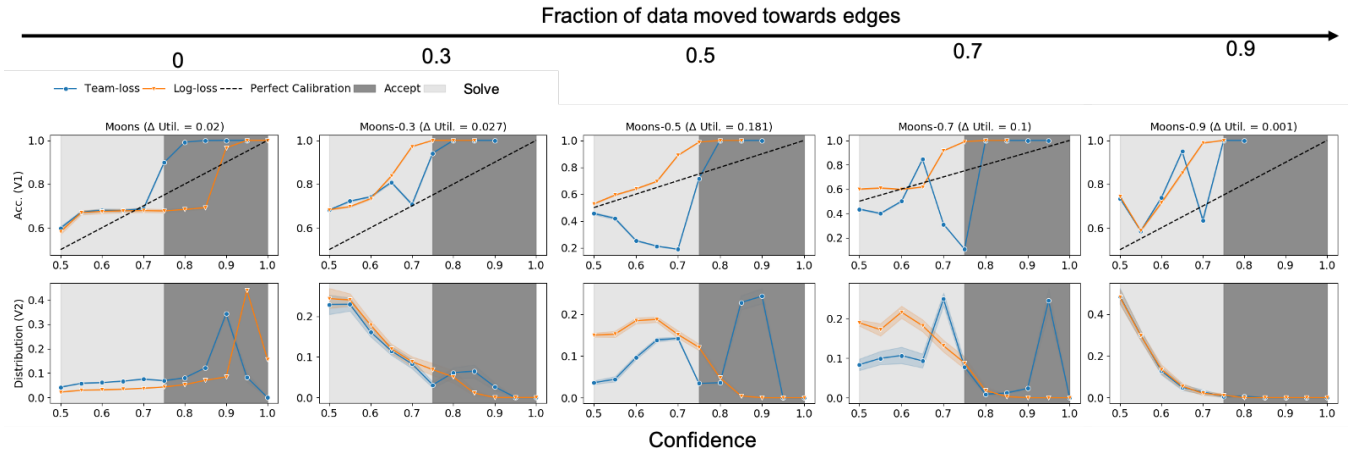


Figure 9: Difference between the predictions of *team-loss* and *log-loss* as we varied the data distribution of the Moons dataset. As we moved more points towards the outer edges of the moons, the behavior of *team-loss* changed from B2 to a combination of B2 and B1; for example, when 50% was moved, *team-loss* both sacrificed accuracy in the `Solve` region and also improved accuracy in `Accept` region.

false-negatives may be costlier than false-positives [Zadrozny *et al.*, 2003]. A common solution here is up-weighting the inputs where the mistakes are costlier. Also relevant is work on *importance-based learning* where re-weighting helps learn from imbalanced data or speed-up training. However, in our setup, re-weighting the inputs makes less sense— the weights would depend on the classifier's output, which has not been trained yet. An iterative approach may be possible, but our initial analysis showed this approach is prone to oscillations, where the classifier may never converge. We leave exploring this avenue for future work.

A fundamental line of work that renders AI predictions more actionable (for humans) and better suitable for teaming is *confidence-calibration*, for example, using Bayesian models [Ghahramani, 2015; Beach, 1975; Gal and Ghahramani, 2016] or via *post-hoc* calibration [Platt, 1999; Zadrozny and Elkan, 2001; Guo *et al.*, 2017; Niculescu-Mizil and Caruana, 2005]. A key difference between these methods and our approach is that *team-loss re-trains* the model to improve on inputs on which users are more likely to rely on the AI predictions. The same contrast distinguishes our approach from *outlier detection* techniques [Hendrycks *et al.*, 2018; Lee *et al.*, 2017; Hodge and Austin, 2004].

More recent work that adjusts model behavior to accommodate collaboration is *backward-compatibility for AI* [Bansal *et al.*, 2019b], where the model considers user interactions with a previous version of the system to preserve trust across updates. Recent user studies showed that when users develop mental models of system's mistakes, properties other than accuracy are also desirable for successful collaboration, for example, *parsimonious* and *deterministic* error boundaries [Bansal *et al.*, 2019a]. Our approach is a first step towards implementing these desiderata within machine learning optimization itself. Other approaches on human-centered optimization regularize or constrain model optimization for other human-centered requirements such as interpretability [Wu *et al.*, 2019; Wu *et al.*, 2018] or fairness [Jung *et al.*, 2019; Zafar *et al.*, 2015].

6 Conclusions

We studied the problem of training classifiers that optimize team performance, a metric that for collaboration matters than mere automation accuracy. To support direct optimization of team performance we advised a new loss function with a formulation based on the expected utility of the human-AI team for decision making. Thorough investigations and visualizations of classifier behavior before and after leveraging *team-loss* for optimization show that, when such an optimization is effective, *team-loss* can fundamentally change model behavior and improve team utility. Changes in model behavior include either (i) sacrificing model accuracy in low confidence regions for more accurate high-confidence predictions, or (ii) increasing accuracy in the `Accept` region through more accurate predictions but fewer highly confident ones. Such behaviors were observed in synthetic and real-world datasets where AI is known to be employed as support for human decision makers. However, we also report that current optimization techniques were not always effective and in fact sometimes they did not change model behavior, i.e., models remain identical even after fine-tuning with *team-loss*. Since *team-loss* clearly emphasizes optimization challenges mostly related to its flat curvature and potential local minimas in the `Solve` region, we invite future work on machine learning optimization and human-AI collaboration to jointly approach such challenges at the intersection of both fields.

7 Acknowledgements

This material is based upon work supported by ONR grant N00014-18-1-2193, the University of Washington WRF/Cable Professorship, the Allen Institute for Artificial Intelligence (AI2), and Microsoft Research. The authors thank Rich Caruana, Bryan Wilder, and Zeyuan Allen-Zhu for useful discussions and comments.

References

[Bansal *et al.*, 2019a] Gagan Bansal, Besmira Nushi, Ece Kamar, Walter S Lasecki, Daniel S Weld, and Eric Horvitz. Beyond accuracy: The role of mental models in human-ai team performance. In *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, volume 7, pages 2–11, 2019.

[Bansal *et al.*, 2019b] Gagan Bansal, Besmira Nushi, Ece Kamar, Daniel S Weld, Walter S Lasecki, and Eric Horvitz. Updates in human-ai teams: Understanding and addressing the performance/compatibility tradeoff. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 2429–2437, 2019.

[Beach, 1975] Barbara Heinrich Beach. Expert judgment about uncertainty: Bayesian decision making in realistic settings. *Organizational Behavior and Human Performance*, 14(1):10–59, 1975.

[Burges, 1998] Christopher JC Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167, 1998.

[Caruana *et al.*, 2015] Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie Elhadad. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *KDD*, 2015.

[Gal and Ghahramani, 2016] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059, 2016.

[GDPR, 2020] GDPR. Art. 22 gdpr, automated individual decision-making, including profiling. https://gdpr-info.eu/art-22-gdpr/, 2020. [Online; accessed 14-January-2020].

[Ghahramani, 2015] Zoubin Ghahramani. Probabilistic machine learning and artificial intelligence. *Nature*, 521(7553):452, 2015.

[Guo *et al.*, 2017] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1321–1330. JMLR. org, 2017.

[Hendrycks and Gimpel, 2018] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv:1610.02136v3*, 2018.

[Hendrycks *et al.*, 2018] Dan Hendrycks, Mantas Mazeika, and Thomas G Dietterich. Deep anomaly detection with outlier exposure. *arXiv preprint arXiv:1812.04606*, 2018.

[Hodge and Austin, 2004] Victoria Hodge and Jim Austin. A survey of outlier detection methodologies. *Artificial intelligence review*, 22(2):85–126, 2004.

[Jung *et al.*, 2019] Christopher Jung, Michael Kearns, Seth Neel, Aaron Roth, Logan Stapleton, and Zhiwei Steven Wu. Eliciting and enforcing subjective individual fairness. *arXiv preprint arXiv:1905.10660*, 2019.

[Kamar *et al.*, 2012] Ece Kamar, Severin Hacker, and Eric Horvitz. Combining human and machine intelligence in large-scale crowdsourcing. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 467–474. International Foundation for Autonomous Agents and Multiagent Systems, 2012.

[Kleinberg *et al.*, 2018] Jon Kleinberg, Himabindu Lakkaraju, Jure Leskovec, Jens Ludwig, and Sendhil Mullainathan. Human decisions and machine predictions. *The quarterly journal of economics*, 133(1):237–293, 2018.

[Lee *et al.*, 2017] Kimin Lee, Honglak Lee, Kibok Lee, and Jinwoo Shin. Training confidence-calibrated classifiers for detecting out-of-distribution samples. *arXiv preprint arXiv:1711.09325*, 2017.

[Nagar and Malone, 2011] Yiftach Nagar and Thomas Malone. Making business predictions by combining human and machine intelligence in prediction markets. Association for Information Systems, 2011.

[Nickelsburg, 2020] Monica Nickelsburg. Washington state passes landmark facial recognition bill, reining in government use of AI. https://www.geekwire.com/2020/washington-state-passes-landmark-facial-recognition-bill-reining-gove 2020.

[Niculescu-Mizil and Caruana, 2005] Alexandru Niculescu-Mizil and Rich Caruana. Predicting good probabilities with supervised learning. In *Proceedings of the 22nd international conference on Machine learning*, pages 625–632. ACM, 2005.

[Patel *et al.*, 2019] Bhavik N Patel, Louis Rosenberg, Gregg Willcox, David Baltaxe, Mimi Lyons, Jeremy Irvin, Pranav Rajpurkar, Timothy Amrhein, Rajan Gupta, Safwan Halabi, et al. Human–machine partnership with artificial intelligence for chest radiograph diagnosis. *NPJ digital medicine*, 2(1):1–10, 2019.

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

[Ribeiro *et al.*, 2016] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why should I trust you?": Explaining the predictions of any classifier. In *Proc. of KDD*, 2016.

[Weld and Bansal, 2019] Daniel S. Weld and Gagan Bansal. The challenge of crafting intelligible intelligence. *Commun. ACM*, 62:70–79, 2019.

[Wu *et al.*, 2018] Mike Wu, Michael C Hughes, Sonali Parbhoo, Maurizio Zazzi, Volker Roth, and Finale Doshi-Velez. Beyond sparsity: Tree regularization of deep models for interpretability. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[Wu *et al.*, 2019] Mike Wu, Sonali Parbhoo, Michael Hughes, Ryan Kindle, Leo Celi, Maurizio Zazzi, Volker

Roth, and Finale Doshi-Velez. Regional tree regularization for interpretability in black box models. *arXiv preprint arXiv:1908.04494*, 2019.

[Zadrozny and Elkan, 2001] Bianca Zadrozny and Charles Elkan. Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In *Icml*, volume 1, pages 609–616. Citeseer, 2001.

[Zadrozny *et al.*, 2003] Bianca Zadrozny, John Langford, and Naoki Abe. Cost-sensitive learning by cost-proportionate example weighting. In *ICDM*, volume 3, page 435, 2003.

[Zafar *et al.*, 2015] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez Rodriguez, and Krishna P Gummadi. Fairness constraints: Mechanisms for fair classification. *arXiv preprint arXiv:1507.05259*, 2015.

8 Appendix

8.1 Extension: Users May Not be Rational

In Section 2 we assumed that the user acted rationally while making the meta-decision. We now relax this assumption and assume that with a small probability $\epsilon \sim B(\gamma_1, \gamma_2)$ the user may (uniformly) randomly choose between `Accept` and `Solve`.[3] Then, extending Equation 4, the user will `Accept` system recommendation with probability:

$$P_\epsilon(m = \texttt{A}) = \begin{cases} 1 - \frac{\epsilon}{2} & \text{if } h(x)[\hat{y}] \geq c(\beta, \lambda, a) \\ \frac{\epsilon}{2} & \text{otherwise} \end{cases} \tag{9}$$

In the above equation, when the model is confident, probability decreased by $\frac{\epsilon}{2}$ because the user may decide to `Solve`. Similarly, when the model is not confident, the $\frac{\epsilon}{2}$ increase (compared to Equation 4) indicates that the user may randomly decide to `Accept` an uncertain recommendation.

To simplify deriving the new equation for the expected utility, we introduce re-write Equation 6 as:

$$\psi(x, y) = \begin{cases} \psi_{\texttt{A}}(x, y) & \text{if } h(x)[\hat{y}] \geq c(\beta, \lambda, a) \\ \psi_{\texttt{S}}(x, y) & \text{otherwise} \end{cases} \tag{10}$$

Using the above two equations, we obtain the following equation for expected utility when the user is not perfectly rational:

$$\psi_\epsilon(x, y) = \begin{cases} (1 - \frac{\epsilon}{2}) \cdot \psi_{\texttt{A}}(x, y) + \frac{\epsilon}{2} \cdot \psi_{\texttt{S}}(x, y) & \text{if } h(x)[\hat{y}] \geq c(\beta, \lambda, a) \\ (1 - \frac{\epsilon}{2}) \cdot \psi_{\texttt{S}}(x, y) + \frac{\epsilon}{2} \cdot \psi_{\texttt{A}}(x, y) & \text{otherwise} \end{cases} \tag{11}$$

The above equation denotes that, when the system is confident, instead of always obtaining $\psi_{\texttt{A}}$ as in Equation 10, with a small probability $\frac{\epsilon}{2}$ the user may obtain the expected utility associated with an `Solve` action. Similarly, when the system is uncertain, the user may sometimes obtain expected utility associated with an `Accept` action. Qualitatively, this will result in a worse best-case expected utility, an artifact of user making sub-optimal decision (to `Solve`) when automation would result in the highest utility. Similarly, the expected utility in the `Solve` region will also decrease– the user may `Accept` uncertain recommendations. On the other hand, this will improve the worst-case utility— the new user will avoid some high-confidence mistakes that a rational user would not. However, unlike ψ, $\psi_{epsilon}$ is strictly monotonic: $\psi_{\texttt{A}}$ is a linear function and hence strictly monotonic, and sum of strictly monotonic and constant function is strictly monotonic.

[3] Note γ_1, γ_2 can be conditioned on confidence and threshold.