Robust Learning from Discriminative Feature Feedback

Sanjoy Dasgupta
Department of Computer Science and Engineering
University of California, San Diego
California, USA

Sivan Sabato
Department of Computer Science
Ben-Gurion University of the Negev
Beer Sheva, Israel

Abstract

Recent work introduced the model of *learning from discriminative feature feedback*, in which a human annotator not only provides labels of instances, but also identifies discriminative features that highlight important differences between pairs of instances. It was shown that such feedback can be conducive to learning, and makes it possible to efficiently learn some concept classes that would otherwise be intractable. However, these results all relied upon *perfect* annotator feedback. In this paper, we introduce a more realistic, *robust* version of the framework, in which the annotator is allowed to make mistakes. We show how such errors can be handled algorithmically, in both an adversarial and a stochastic setting. In particular, we derive regret bounds in both settings that, as in the case of a perfect annotator, are independent of the number of features. We show that this result cannot be obtained by a naive reduction from the robust setting to the non-robust setting.

1 Introduction

There has been a growing interest in learning from data sets in which instances not only have labels but may also have some information about relevant features. One way to think about this is that the human annotator labels each instance and also tries to pick out one or two features of the instance that help to (weakly) explain this label. The hope is that this will (1) lead to better models being learned, (2) reduce the number of instances needed for learning, and (3) help pave the way for more explainable models.

Proceedings of the 23[rd]International Conference on Artificial Intelligence and Statistics (AISTATS) 2020, Palermo, Italy. PMLR: Volume 108.

For instance, early work in information retrieval (Croft and Das, 1990) looked at a simple protocol in which a user who labels a document (as, say, "sports") also highlights one or two words (like "goalie") that are predictive of this label. Such feedback is not very costly, since the labeler is in any case reading the document, but can be very helpful with identifying relevant features in the high-dimensional space of words. Numerous variations of this idea have been explored for text and vision applications (Croft and Das, 1990; Raghavan et al., 2005; Druck et al., 2008; Settles, 2011; Mac Aodha et al., 2018). Some theoretical studies (Poulis and Dasgupta, 2017; Visotsky et al., 2019) have also formalized such schemes and shown that, in some situations, they lead to markedly better sample complexity than would be achieved when learning from labels alone.

Another type of feature feedback, which has been explored in human-in-the-loop computer vision work (Branson et al., 2010; Zou et al., 2015), asks the human to provide features that *distinguish* between two instances: for instance, the feature "stripes" distinguishes a zebra from a horse. The idea is that this is more concrete than suggesting predictive features and might thus be easier for the annotator to do reliably, especially in a multi-class setting. A formal model of this process was recently suggested by Dasgupta et al. (2018). In this protocol, termed *discriminative feature feedback*, learning takes place in rounds of interaction, where in each round the learner makes a prediction on the current example, and provides a previous example as an "explanation". If the prediction is incorrect, the teacher provides the correct prediction, and a feature distinguishing the incorrect explanation from the current example. The precise protocol and its semantics are reviewed in Section 2. The work of Dasgupta et al. (2018) provides a learning algorithm that uses this type of discriminative feedback and gives a mistake bound for it. Interestingly, the richer feedback makes it possible to learn some concept classes, such as DNF (disjunctive normal form, OR-of-AND) formulas, that are known to be computationally hard to learn from labels alone.

However, a significant drawback of that work is that it assumes that the human teacher never makes mistakes when labeling points or providing discriminative features. This is unrealistic in practice. In this paper, we introduce a *robust* discriminative feature feedback setting, and provide two robust algorithms for learning in this setting. The first algorithm considers a fixed data set that contains some "exceptions": points on which the teacher can make arbitrary errors. If, for example, the learning task is to distinguish between mammals, reptiles, amphibians, and so on, then these exceptions might be animals like penguin or platypus, corner cases that tend to defy simple rules. The second algorithm is for a statistical setting in which points are drawn i.i.d. from some underlying distribution, and a constant fraction of them are exceptions. In both cases, we provide proofs of correctness and mistake bounds.

Our contributions. Our first contribution (Section 3) is to formulate a noise model for discriminative feature feedback that allows the teacher to behave arbitrarily on some subset of instances.

Second, we show that although the work of Dasgupta et al. (2018) could, in principle, handle these exceptions by treating them as correct and devising more complicated rules to accommodate them, this would result in a large increase in the complexity of the concepts being learned (Theorem 1). This, in turn, would lead to a large number of mistakes on the data set. To complete the argument, we provide a new lower bound on the best mistake bound obtainable in the perfect-annotation setting, as a function of representation size (Theorem 2). In particular, we show that if the number of features is unbounded, as allowed by the original discriminative feature feedback setting, then this attempt to handle mistakes leads to a vacuous mistake bound.

Finally, we provide two new algorithms for robust learning under discriminative feature feedback, first in an adversarial setting where the ordering of instances is worst-case (Section 5), and then in a stochastic setting where the instances are sampled from an underlying distribution (Section 6). In both cases, we provide mistake bounds in terms of the size of the concept being learned and the number, or fraction, of exceptions (Theorems 3 and 9), but without any dependence on the number of features.

2 Preliminaries

Dasgupta et al. (2018) defined the discriminative feature feedback model and studied it in a perfect-annotation setting. Let c^* be the target concept to be learned, where c^* is a mapping from the input space $\mathcal{X}$ to a finite label space $\mathcal{Y}$. The learner has access to a set of Boolean features Φ on $\mathcal{X}$, and expresses concepts in terms of these. It is assumed that $\mathcal{X}$ can be represented as the union of m sets in some family of sets $\mathcal{G} = \{G_1, \ldots, G_m\}$, $\mathcal{X} = G_1 \cup G_2 \cup \cdots \cup G_m$. This is the *internal representation* of the teacher. The representation, which is unknown to the learner, satisfies the following properties:

- Each of the sets is pure in its label: for each i, there exists a label $\ell(G_i) \in \mathcal{Y}$ such that $\forall x \in G_i, c^*(x) = \ell(G_i)$
- Any two sets G_i, G_j with $\ell(G_i) \neq \ell(G_j)$ have a *discriminating feature*: there is some $\phi \in \Phi$ such that if $x \in G_i$, $\phi(x)$ is satisfied, and if $x \in G_j$, $\phi(x)$ is not satisfied.

No restrictions are placed on the number of possible features, which can even be infinite. Therefore, negations and logical combinations of features can also be used as discriminative features.

For any $x \in \mathcal{X}$, denote by $G(x) \in \mathcal{G}$ some set containing x. If there are multiple such components, $G(x)$ is some fixed choice. The interactive learning protocol for the noiseless model is as follows:

- A new instance x_t arrives.
- The learner supplies a prediction $\widehat{y}_t$, and an instance $\widehat{x}_t$ which was previously seen with that label ("an explanation").
- If the prediction is correct, no feedback is obtained.
- If the prediction is incorrect, the teacher provides the correct label $y_t = c^*(x)$, and a feature ϕ that separates $G(x_t)$ from $G(\widehat{x}_t)$, that is

$$\phi(x) = \begin{cases} \texttt{true} & \text{if } x \in G(x_t), \\ \texttt{false} & \text{if } x \in G(\widehat{x}_t). \end{cases}$$

Here, a feature is any mapping from examples to `true`/`false`, which can either be given explicitly as a coordinate of x, or be calculated from its representation. It is shown in Dasgupta et al. (2018) that a legal representation of size m exists if and only if the concept c^* can be represented by a DNF formula of a special form, which they call a "separable-DNF". Dasgupta et al. (2018) give an algorithm for this interaction model, which obtains a mistake bound of m^2, with no dependence on the number of available features $|\Phi|$.

3 A feedback model with mistakes

In this work, we propose an extension of the discriminative feature feedback model to a model that allows

mistakes. First, note that any deterministic labeling function (that is, one in which the same example always gets the same label) can be modeled in the perfect-annotation model described above, since one can always model $\mathcal{G}$ as a set of singletons, one for each example in the input stream. However, this is clearly unhelpful, as there can be no generalization to unseen examples, and the number of mistakes that the algorithm makes cannot be bounded. In particular, the mistake bound of m^2 obtained in Dasgupta et al. (2018) is meaningless if m is equal to the number of examples. In fact, as we show in Sec. 4, even a small number of adversarial changes to a perfect model can lead to an unreasonably large representation.

We thus propose to allow a trade-off between the number of components modeling the concept and the number of *exceptions*, which are examples that deviate from the model. In this setting, we assume as above that there are m components. However, instead of requiring that for all $x \in G_i$, $c^*(x) = \ell(G_i)$, we allow some exceptions. Formally, let

$$M = M(c^*, \mathcal{G}) := \{x \in \mathcal{X} \mid c^*(x) \neq \ell(G(x))\}. \quad (1)$$

This is the set of exceptions which deviate from the representation $G_1, \ldots, G_m$. If the teacher provides a discriminative feature between a pair of examples that includes at least one exception, the feature might not be one that discriminates the respective components. In all other cases, the teacher behaves as in the perfect-annotation setting.

We study two cases: one in which the input is adversarial and $|M|$ is upper-bounded by some integer, and one in which the stream is an i.i.d. draw from a distribution and the probability mass of M is upper-bounded by some small value. An additional parameter that we consider is related to the amount of consistency among exceptions. Formally, for an example $\widehat{x} \notin M$ and a feature ϕ, define

$$M_{\widehat{x},\phi} = \{x \in M \mid \phi \text{ is returned as the discriminating feature between } x \text{ and } G(\widehat{x})\}. \quad (2)$$

For any $\widehat{x}, \phi$, we have $M_{\widehat{x},\phi} \subseteq M$, thus one can always upper-bound $M_{\widehat{x},\phi}$ using the size of M. However, in many cases it is more reasonable to assume that different exceptions would not generally use the same discriminative features, for instance if the exceptions are not the result of a coordinated corruption. Thus, we set a separate upper bound on $\max_{\widehat{x} \notin M, \phi} |M_{\widehat{x},\phi}|$, which can be significantly smaller than $|M|$.

We make an additional technical assumption, which was not explicitly assumed in Dasgupta et al. (2018) where a perfect annotator was assumed: If the same two components are separated by the teacher more than once during the whole interaction with the learner, then the same feature is provided in all of these interactions. Note that this requirement is always satisfied by some representation, if examples separated by different features are allocated to different components.

To conclude the definition of the setting, observe that on top of exceptions as defined above, the teacher can deviate from the interactive protocol in other ways. For instance, it can provide a feature ϕ that does not actually separate the two provided examples, or it can flag the same label on the same example first as a correct label and then later as a wrong label, violating the assumption of a deterministic labeling function. However, these types of inconsistencies can be easily identified when the feedback is provided, and ignored by the learner. Thus, for simplicity, we assume below that no such inconsistencies occur. Another type of deviation from the protocol can occur if the teacher provides a feature that does not actually separate the two components $G(x)$ and $G(\widehat{x})$ (although it does separate x and $\widehat{x}$). This type of exception can be handled the same as exceptions in M. In summary, all exceptions are either easy to identify immediately, or covered by the current exception model.

4 Exceptions under the perfect-annotation model

As discussed above, any deterministic labeling, including one with exceptions as defined above, can be modeled by the perfect-annotation setting, for instance by creating a special group G_i for each exception, and dissecting other groups to make sure that the discriminative-feature property holds. In this section, we show that nonetheless, attempting to reduce a model with mistakes to a perfect-annotation model can result in a very large mistake bound when the number of possible features is large. First, we provide upper and lower bounds on the number of components required for such a reduction.

By a *representation* $\mathcal{G}$, we mean a family of sets $\mathcal{G} = \{G_1, G_2, \ldots\}$ that cover $\mathcal{X}$ and a labeling $\ell(G_i)$ of each set. The *size* of the representation is $|\mathcal{G}|$. Recall from (1) that $M(c, \mathcal{G})$ denotes the set of exceptions for a given concept c and representation $\mathcal{G}$.

Theorem 1. *Let $\mathcal{G}$ be a representation of size m. Let $\bar{c}$ be a concept with k exceptions, that is $|M(\bar{c}, \mathcal{G})| = k$. Let $\bar{\mathcal{G}}$ be a representation of a minimal size $\bar{m}$ such that $|M(\bar{c}, \bar{\mathcal{G}})| = 0$. Let $d = |\Phi|$ be the number of available features. Then:*

(a) $\bar{m} \leq m + dk$.

(b) There exists a case in which $m = 1$ while $\bar{m} \geq d+1$.

The proof is provided in the supplementary material. We remark that the bound in the theorem above is intimately related to the *DNF exception problem*, which studies how many clauses are required to represent a concept defined by a DNF of a certain size with a bounded list of exceptions. This problem has been studied in several works (Zhuravlev, 1985; Kogan, 1987; Mubayi et al., 2006; Maximov, 2013), including in the context of active learning with membership queries (Angluin and Krikis, 1994; Angluin et al., 1997); however, tight upper and lower bounds are not known for this problem.

What is the significance of the representation size? The algorithm of Dasgupta et al. (2018) for the perfect-annotation setting makes $\Theta(m^2)$ mistakes, where m is the representation size. However, they do not answer the question whether the order of this mistake bound is optimal. The following lower bound shows that it is, implying that the representation size is a crucial property. In particular, combined with Theorem 1, it follows that reducing the setting which allows mistakes to the perfect-annotation setting when the number of features is unbounded would result in a vacuous mistake bound.

Theorem 2. *If feature feedback is given with respect to a representation of size m, then any algorithm must have a mistake bound $\Omega(m^2)$ in the perfect-annotation setting.*

The proof is provided in the supplementary material. We have thus shown that a reduction of the setting with mistakes to the perfect-annotation setting results in a mistake bound that depends on the number of features d, which can be unbounded. In the next section we propose a robust algorithm which allows mistakes, and obtains an improved mistake bound, which does not depend on d.

5 Robust feature feedback in an adversarial setting

In this section, we derive a robust algorithm under an adversarial model. In this model, there are no limitations on the input stream except that it conforms to the interaction protocol described in Sec. 3. In particular, the exceptions can appear at any arbitrary location in the stream. We assume that the number of exceptions (the size of M) is upper-bounded by k for some integer k, and that for any $\widehat{x} \notin M$ and any ϕ, $|M_{\widehat{x},\phi}| \leq s$ for some integer $s \leq k$; recall the definitions (1) and (2). We say that s is an upper bound on the number of *similar* exceptions. We propose an algorithm for this setting, called `RobustDFF`, and derive the following mistake bound for this algorithm.

Theorem 3. *If there is a representation of size at most m which satisfies the bounds of k and s defined above, then the number of mistakes made by `RobustDFF` is at most $(m+k)((s+1)(m-1)+k+2)$, which is $O\big(((s+1)m+k)\cdot(m+k)\big)$.*

Note that for $k = s = 0$, we retrieve the optimal mistake bound order of $O(m^2)$ for the perfect-annotation setting. Setting $s = k$ obtains a mistake bound of $O(km(m+k))$. Comparing this upper bound with the conclusions from Theorem 1 for the case $s = k$, it can be seen that a reduction to the perfect-annotation setting leads to a mistake bound of $O(m+dk)^2$. Thus, if $d \gg m$ then the mistake bound of `RobustDFF` is preferable. Below, we present the algorithm and the mistake-bound analysis.

5.1 Robust algorithm for the adversarial setting

Algorithm 1 `RobustDFF`: Robust discriminative feature feedback for the adversarial setting

Input: Max. components m, max. exceptions k, max. similar exceptions $s \leq k$.

$t \leftarrow 0$
Get the label y_o of the first example x_o
Initialize L to an empty list
while true **do**
  $t \leftarrow t+1$
  get a new point x_t:
  if $\exists C[\widehat{x}] \in L$ such that x_t satisfies $C[\widehat{x}]$ **then**
    Predict `label`$[\widehat{x}]$ and provide example $\widehat{x}$
    if prediction is incorrect **then**
      Get correct label y_t and feature ϕ
      Update `fcount`$[\widehat{x}]$, $C[\widehat{x}]$, L by running:
        `HandleMistake`$(m, \widehat{x}, k, s, \phi)$ (Alg. 2).
    end if
  else (no relevant rule exists)
    Predict y_0 and provide example x_0
    if prediction is incorrect **then**
      Get correct label y_t and feature ϕ.
      Add to L an empty conjunction $C[x_t]$,
        and set `label`$[x_t] \leftarrow y_t$.
      Initialize `fcount`$[\widehat{x}](\cdot)$ to 0.
    end if
  end if
end while

`RobustDFF` is listed in Alg. 1. It calls the procedure `HandleMistake`, given in Alg. 2. The algorithm maintains a set of conjunctions (rules) which are iteratively refined based on the feedback from the teacher. A rule is *created* if an example that matches none of the existing conjunctions appears. A rule is *refined* if mistakes with the feedback from the teacher warrants such a

Algorithm 2 `HandleMistake`: Handling an incorrect prediction for a given rule

Input: Max. components m, max. exceptions k, max. similar exceptions $s \leq k$, rule representative $\widehat{x}$, discriminating feature ϕ, access to $\texttt{fcount}, C, L$
Output: Updates values of $\texttt{fcount}[\widehat{x}], C[\widehat{x}], L$

Add 1 to $\texttt{fcount}[\widehat{x}](\phi)$
if $\texttt{fcount}[\widehat{x}](\phi) > s$ **then**
  $C[\widehat{x}] \leftarrow C[\widehat{x}] \wedge \neg\phi$
  $\texttt{fcount}[\widehat{x}](\phi) \leftarrow 0$
  if $|C[\widehat{x}]| \geq m$ **then**
    delete $C[\widehat{x}]$ from L
  end if
else
  $b \leftarrow m - 1 - |C[\widehat{x}]|$
  if the sum of counters $\texttt{fcount}[\widehat{x}](\phi)$ for all ϕ except for the b largest counters is more than k **then** remove $C[\widehat{x}]$ from L.
end if

refinement. A rule may also be deleted.

`RobustDFF` keeps track of the following information:

- The first labeled example (x_0, y_0).
- A list of conjunctions L.
- For every conjunction $C[x] \in L$, its label, denoted $\texttt{label}[x]$
- For every conjunction $C[x] \in L$, a mapping $\texttt{fcount}[x] : \Phi \to \mathbb{N}$ of counters, which count, for each feature, how many times it was provided by the teacher as a discriminating feature for x. Since Φ might not be finite, $\texttt{fcount}[x](\phi)$ is only explicitly set when the counter is incremented for the first time. All uninitialized counters are treated as having a value of zero.

Exceptions might cause issues in rules in one of two ways: either a rule is created based on an exception, or it is wrongly refined based on one. To avoid the latter, a rule based on a non-exception is only refined when there is at least one non-exception that warrants this specific refinement. This is guaranteed by collecting more than s witnesses to a certain feature, before deciding on a rule refinement based on this feature. Creating rules based on exceptions is not prevented in `RobustDFF`. Instead, the algorithm identifies rules that become too large, or have too many separating features, and removes them. We show in the analysis that this upper-bounds the number of mistakes that the algorithm makes due to rules based on exceptions, while keeping good rules intact.

5.2 Mistake bound for the adversarial setting

We now prove Theorem 3, the mistake bound of `RobustDFF`. We first prove several invariants of the algorithm. First, we prove that in rules representing components, these components are never split.

Lemma 4. *At all times in the algorithm, if $\widehat{x}$ is not an exception then conjunction $C[\widehat{x}]$ is satisfied by every point in $G(\widehat{x})$. In addition, for every literal ϕ in $C[\widehat{x}]$, there is some non-exception x such that $G(x)$ is separated from $G(\widehat{x})$ by ϕ.*

Proof. We prove the claim by induction on the length of $C[\widehat{x}]$. When $C[\widehat{x}]$ is first created, it is an empty conjunction so it is satisfied by all of $G(\widehat{x})$. When $C[\widehat{x}]$ is restricted by $\neg\phi$ in `HandleMistake`, it means that $s+1$ examples were separated from $\widehat{x}$ by ϕ. By the assumption that $|M_{\widehat{x},\phi}| \leq s$, it follows that at least one of these examples, call it x, is not an exception, hence $G(\widehat{x})$ is separated from $G(x)$ by ϕ. This implies that $G(\widehat{x})$ has no examples that are satisfied by ϕ. Hence, after adding $\neg\phi$ to $C[\widehat{x}]$, the extended $C[\widehat{x}]$ is still satisfied by $G(\widehat{x})$ and is separated by ϕ from $G(x)$. □

Next, we prove that two rules never represent the same component.

Lemma 5. *For any two non-exceptions x, x', if there are two rules $C[x]$ and $C[x']$ in L then $G(x) \neq G(x')$.*

Proof. Suppose x is observed earlier in the input sequence and x' is observed later; If $C[x]$ is generated and $C[x']$ is also generated, this means that $C[x]$, in its form when x' is observed, does not satisfy x'. But by Lemma 4, $C[x]$ always satisfies $G(x)$. Hence, $x' \notin G(x)$, which implies the claim. □

Next, we prove that only rules created by exceptions might be deleted.

Lemma 6. *If `HandleMistake` when run by `RobustDFF` deletes the rule $C[\widehat{x}]$, then $\widehat{x}$ is an exception.*

Proof. Assume for contradiction that $\widehat{x}$ is not an exception but rule $C[\widehat{x}]$ is deleted. A rule can get deleted for one of two reasons. The first reason for deletion is if the conjunction $C[\widehat{x}]$ has at least m literals. Then, by Lemma 4, for each such literal in $C[\widehat{x}]$ there is some non-exception x such that $G(x)$ is separated from $G(\widehat{x})$ using that literal. Since there are m components G_i, there are at most $m-1$ literals in $C[\widehat{x}]$, which is a contradiction to the size of $C[\widehat{x}]$. The second reason for deletion is if the sum of the counters $\texttt{fcount}[\widehat{x}](\phi)$ except for the largest $b \equiv m - |C[\widehat{x}]| - 1$ counters is more than k. Suppose that $\widehat{x}$ is not an exception. By Lemma 4, $|C[\widehat{x}]|$ components are already separated

from it using literals in $C[\widehat{x}]$. At most b other components could have some overlap with $C[\widehat{x}]$. Thus, at most b of the non-zero counters $\texttt{fcount}[\widehat{x}](\phi)$ have a ϕ which separates $G(\widehat{x})$ from some component that has an overlap with $C[\widehat{x}]$. All other counters must have been generated by exceptions, and the total number of such exceptions is at least the sum of the other counters. By the condition for deleting a rule, more than k such exceptions were observed. But this contradicts the upper bound of k for exceptions.

In both cases, we reached a contradiction. Hence, $\widehat{x}$ is an exception. □

To bound the total number of mistakes, we first bound the total number of rules created by the algorithm.

Lemma 7. *`RobustDFF` creates at most $m + k$ rules.*

Proof. By Lemma 5, the total number of rules in L generated by non-exceptions is at most the number of components, m. Therefore, at most m non-exception rules are ever generated. By Lemma 6, only rules generated by exceptions might be deleted. Since rules are generated at most once for every input example, and there are at most k exceptions in the input, at most k rules generated by exceptions are ever generated. □

Next, we bound the number of mistakes associated with each rule.

Lemma 8. *The number of mistakes resulting from examples that have been matched to a single rule $C[x]$ is at most $(s + 1)(m - 1) + k + 1$.*

Proof. For all x, ϕ, at the end of each round of `RobustDFF`, $\texttt{fcount}[x](\phi) \leq s$, since each new mistake that is matched to $C[x]$ increases some $\texttt{fcount}[x](\phi)$ by 1, and then, if $\texttt{fcount}[x](\phi) = s + 1$, zeros this counter and extends $C[x]$ by one. Therefore, for every feature that end up extending $C[x]$, there are at most $s + 1$ mistakes on $C[x]$. Letting r be the length of $C[x]$ after the last iteration in which it exists, this means that exactly $(s + 1)r$ mistakes are matched with features that extend $C[x]$.

The number of mistakes that do not match features that extend $C[x]$ is always at most $k+s(m-1-|C[x]|)$ at the end of an iteration, since if at any time during the run the sum of counters is increased beyond this number, it means that the sum of the counters except for the $m - 1 - |C[x]|$ largest ones is $k + 1$, in which case the rule gets deleted. Also, whenever the rule is extended, one counter with value s is zeroed, thus this property continues to hold. Thus, the total number of mistakes for $C[x]$ is at most $(s+1)r+k+1+s(m-1-r) \leq s(m-1)+r+k+1$, Since $r \leq m-1$, this proves the claim. □

Theorem 3 is now immediate, as follows: Each rule makes at most $(s + 1)(m - 1) + k + 1$ mistakes by Lemma 8. By Lemma 7, at most $m + k$ rule are generated by `RobustDFF`. In addition, a mistake that does not match any rule creates a new rule, thus there are at most $m + k$ such mistakes. In total, `RobustDFF` makes at most $(m + k)((s + 1)(m - 1) + k + 2)$ mistakes.

This concludes the analysis of the adversarial robust algorithm. In the next section, we study a robust algorithm for a stochastic setting.

6 Robust feature feedback in a stochastic setting

In this section, we assume that the stream is drawn from a stochastic source, with a probability of at most ϵ that a drawn example is an exception. In addition, we assume that for all non-exceptions $\hat{x}$ and features ϕ, the probability mass of $M_{\widehat{x},\phi}$ is at most $\sigma \leq \epsilon$. The algorithm gets an additional confidence parameter δ as input, and guarantees are provided with a probability of $1 - \delta$.

For a stream of a given size n, it is possible to apply Theorem 3 with $k \approx \epsilon n$ and $s \approx \sigma n$ to get a mistake bound for the stochastic setting. However, the resulting bound grows quadratically with the stream size, rendering it vacuous. Thus, we propose a different algorithm, called `StRoDFF`, and show that for this algorithm, the rate of mistakes for large stream sizes is bounded. We prove the following theorem.

Theorem 9. *Let $\delta \leq 1/e^2$. Suppose that the exception rate is at most $\epsilon \leq \frac{1}{4}$ and let the length of the stream of examples be n. With a probability at least $1 - \delta$, the rate of mistakes of `StRoDFF` on a stream of size n is upper bounded by*

$$O\Big((\sigma m + \epsilon)m \log(1/\delta) + m^2 \log^2(n/\delta)/\sqrt{n}\Big).$$

6.1 Robust algorithm for the stochastic setting

`StRoDFF` is presented in Alg. 3. The structure of `StRoDFF` is similar to that of `RobustDFF`, but some adaptations are required to take advantage of the stochastic assumption. The following additional information is stored by `StRoDFF`: t_{lr} records the last time that a new rule was created. N_{lr} counts the number of examples that were not satisfied by a rule since round t_{lr}. $t(\widehat{x})$ records the time that rule $C[\widehat{x}]$ was created, and $t(\widehat{x}, \phi)$ records the first time that an example with a discriminative feature ϕ was provided for the rule $C[\widehat{x}]$. In addition, `StRoDFF` uses the following

functions:

$$q(\epsilon, t) := \epsilon t + \frac{2}{3}\log(8t^3/\delta) + \sqrt{2\epsilon t \log(8t^3/\delta)}, \quad (3)$$

$$\gamma(\epsilon, r, t) := \frac{1}{1-2\epsilon}(r + 4\sqrt{r}\log^{3/2}(\frac{8t^2}{\delta})) - r + 1. \quad (4)$$

These functions are used to calculate exception thresholds, in place of k and s that are used in `RobustDFF`.

A main difference between `RobustDFF` and `StRoDFF` is that in `StRoDFF`, not every example which is not satisfied by current rules causes the creation of a new rule. Instead, a rule is created only if a specific condition is met (see line 21). This condition compares the number of examples that fell outside L since the last creation of a rule, to the number of examples that fell inside the rules. It is used to guarantee that rules are only created if there is sufficient probability mass outside current rules, thus bounding the number of rules created by exceptions.

Algorithm 3 `StRoDFF`: Robust discriminative feature feedback for the stochastic setting

Input: Max. components m, max. prob. of exceptions ϵ, max. prob. of similar exceptions σ, confidence δ

1: $t \leftarrow 0$; $N_{\text{lr}} \leftarrow 0, t_{\text{lr}} \leftarrow 0$.
2: Get the label y_o of the first example x_o;
3: Initialize L to an empty list
4: **while** true **do**
5: $t \leftarrow t+1$; get a new point x_t.
6: **if** $\exists C[\widehat{x}] \in L$ such that x_t satisfies $C[\widehat{x}]$ **then**
7: Predict $\texttt{label}[\widehat{x}]$ and provide example $\widehat{x}$
8: **if** prediction is incorrect **then**
9: Get correct label y_t and feature ϕ
10: **if** $\texttt{fcount}[\widehat{x}](\phi) = 0$, **then** $t(\widehat{x}, \phi) \leftarrow t$.
11: $t' \leftarrow t - t(\widehat{x}, \phi) + 1$.
12: $n_s \leftarrow q(\sigma, t') + 1$, $n_k \leftarrow q(\epsilon, t')$.
13: Update $\texttt{fcount}[\widehat{x}]$, $C[\widehat{x}]$, L by running:
14: $\texttt{HandleMistake}(m, \widehat{x}, n_k, n_s, \phi)$.
15: **end if**
16: **else** (no relevant rule exists)
17: Predict y_0 and provide example x_0
18: $N_{\text{lr}} \leftarrow N_{\text{lr}} + 1$
19: **if** prediction is incorrect **then**
20: Get correct label y_t and feature ϕ.
21: **if** $N_{\text{lr}} \geq \gamma(\epsilon, t - t_{\text{lr}} - N_{\text{lr}} + 1, t)$ **then**
22: Add to L an empty conj. $C[x_t]$,
23: and set $\texttt{label}[x_t] \leftarrow y_t$.
24: Initialize $\texttt{fcount}[\widehat{x}](\cdot)$ to 0.
25: $t(\widehat{x}) \leftarrow t$, $N_{\text{lr}} \leftarrow 0$, $t_{\text{lr}} \leftarrow t$.
26: **end if**
27: **end if**
28: **end if**
29: **end while**

6.2 Error bound for the stochastic setting

In this section, we prove Theorem 9. First, we define the following events, which together guarantee the correctness of estimates based on $q(\cdot, \cdot)$ in the algorithm.

- $\xi_1 :=$ { At any time t in `StRoDFF`, for any $t' \leq t$, the number of exceptions observed in the last t' iterations is at most $q(\epsilon, t')$. }.
- $\xi_2 :=$ { At any time t in `StRoDFF`, for any $t' \leq t$, if in round $t - t' + 1$ a mistake was made and a feature ϕ separating $\widehat{x}$ was provided by the teacher, then the number of exceptions in $M_{\widehat{x},\phi}$ observed afterwards, until iteration t (inclusive), is at most $q(\sigma, t')$. }.

By Bernstein's inequality and a union bound on all the pairs $t' \leq t$, setting $\delta(t', t) := \delta/(4t^3)$, we get that $\xi = \xi_1 \wedge \xi_2$ holds with a probability at least $1 - \delta/2$.

The proof of Theorem 9 is based on several lemmas. Some of the analysis is analogous to that of `RobustDFF`. However, upper-bounding the number of generated rules requires a new statistical analysis. We first give the lemmas that have direct analogs in the analysis of `RobustDFF`. The following lemma is analogous to Lemma 4.

Lemma 10. *Assume ξ. At all times during the run of `StRoDFF`, if $\widehat{x}$ is not an exception then $C[\widehat{x}]$ is satisfied by every point in $G(\widehat{x})$. In addition, for every literal ϕ in $C[\widehat{x}]$, there is some non-exception x such that $G(x)$ is separated from $G(\widehat{x})$ by ϕ.*

Proof. The proof follows the same argument as the proof of Lemma 4, except that in `StRoDFF`, instead of waiting for $s+1$ examples, `HandleMistake` restricts $C[\widehat{x}]$ by $\neg\phi$ if more than n_s examples were separated from $\widehat{x}$ by ϕ, where $n_s = q(\sigma, t - t(\hat{x}, \phi) + 1) + 1$. By ξ_2, the number of exceptions in $M_{\widehat{x},\phi}$ encountered since the first such example, which was encountered in round $t(\widehat{x}, \phi)$, is at most n_s. Therefore, at least one of the examples separated by ϕ is not an exception. The rest of the proof remains the same as the proof of Lemma 4. □

The following lemma is analogous to Lemma 5, proved above for `RobustDFF`.

Lemma 11. *Assume ξ. In `StRoDFF`, for any two non-exceptions x, x', if there are two rules $C[x]$ and $C[x']$ in L then $G(x) \neq G(x')$.*

Proof. The proof is identical to the proof of Lemma 5, except that it uses Lemma 10 instead of Lemma 4. □

The following lemma is analogous to Lemma 6, proved above for RobustDFF.

Lemma 12. *Assume ξ. In StRoDFF, if a rule $C[\widehat{x}]$ gets deleted then $\widehat{x}$ is an exception.*

Proof. The proof is the same as that of Lemma 6, except that Lemma 10 is used instead of Lemma 4. In addition, instead of the upper bound of k on the number of exceptions which is used by HandleMistake when running from RobustDFF, in the case of StRoDFF the upper bound in HandleMistake on the maximal number of exceptions is set to $n_k := q(\epsilon, t - t(\widehat{x}) + 1)$. Thus, if the sum of the counters $\texttt{fcount}[\widehat{x}](\phi)$ except for the largest $b := m - |C[\widehat{x}]| - 1$ counters is more than n_k, then more than $n_k + 1$ exceptions were observed since the creation of the rule $C[\widehat{x}]$ at time $t(\widehat{x})$, which contradicts ξ. The rest of the proof is identical. □

In the next lemma, it is shown that rules are not created unless there is a significant probability mass outside the current rules. The proof of this lemma is provided in the supplementary material. The main idea of the proof is to show that the condition on line 21 does not hold unless there is a sufficient probability mass outside the current set of rules. This is shown via a suitable concentration inequality, combined with an analysis of the dynamics of rule refinements in StRoDFF.

Lemma 13. *Assume $\epsilon < \frac{1}{4}$ and $\delta \leq 1/e^2$. With a probability at least $1 - \delta/4$, all the rules generated by StRoDFF satisfy the following property: The probability mass of examples that fall outside of L at the time the new rule is created is at least 2ϵ.*

The next lemma upper-bounds the number of rules generated by StRoDFF. Crucially, unlike the case of RobustDFF, this number does not depend on the total number of exceptions, which is linear in the size of the stream in the stochastic setting.

Lemma 14. *Assume $\epsilon < \frac{1}{4}$ and $\delta \leq 1/e^2$. With a probability at least $1 - \delta$, the total number of rules created by the algorithm is at most $R(m, \delta) := 4m \log(4/\delta)$.*

Proof. Assume that ξ holds, which occurs with probability at least $1 - \delta/2$. By Lemma 11 the total number of rules in L generated by non-exceptions is at most the number of components, m. Therefore, at most m non-exception rules are ever generated. To bound the number of rules created based on exceptions, we bound the probability, conditioned on a prefix of the stream, that the next rule created by StRoDFF after processing this prefix, is based on an exception. We use Lemma 13, which shows that with a probability at least $1 - \delta/4$, a rule is created by StRoDFF only if the probability mass of examples that are not satisfied by any of the current rules is at least 2ϵ. Denote the event that the property in Lemma 13 holds by ξ_3.

Under ξ_3, given that an example creates a new rule in round t, this is a random example from the set of examples not satisfied by the current set of rules L. Since the probability mass of exceptions is at most ϵ, and the probability mass outside L is at least 2ϵ, it follows that any new rule has a probability of at most a $\frac{1}{2}$ to be based on an exception. Therefore, under ξ_3, the number of rules created until the next non-exception rule is created is an independent geometric random variable with a success probability of at least a $\frac{1}{2}$. Moreover, at most m rules are created based on non-exceptions. By Lemma 15, which is provided in the supplementary material, the probability that more than $R(m, \delta) := 4m \log(4/\delta)$ trials are required to obtain m non-exception rules is less than $\delta/4$. Applying a union bound along with ξ_3 and ξ, the overall probability that this occurs is at least $1 - \delta$. □

The mistake bound for StRoDFF can now be proved. The proof is provided in the appendix in the supplementary material.

7 Conclusion

Discriminative feature feedback is a promising setting, which allows a more natural learning from a knowledgeable teacher. In this work, we showed that it is possible to learn with discriminative feature feedback even when the annotator is not perfect, and proved mistake bounds that do not depend on the number of features. We note that while the proposed algorithms require the problem parameters as inputs, this can be avoided by using a wrapper algorithm which searches for good parameter values. We defer the details to the long version of this work. The study of learning with rich feedback has the potential to be applicable to many real-life scenarios. In this work we have made an important step towards this goal.

Acknowledgements

This research was supported by National Science Foundation grant CCF-1813160, and by a United-States-Israel Binational Science Foundation (BSF) grant no. 2017641. Part of the work was done while the authors were at the "Foundations of Machine Learning" program at the Simons Institute for the Theory of Computing, Berkeley.

References

D. Angluin and M. Krikis. Learning with malicious membership queries and exceptions. In *Proceedings*

of the seventh annual conference on Computational learning theory, pages 57–66. ACM, 1994.

D. Angluin, M. Kriķis, R. H. Sloan, and G. Turán. Malicious omissions and errors in answers to membership queries. *Machine Learning*, 28(2-3):211–255, 1997.

S. Branson, C. Wah, B. Babenko, F. Schroff, P. Welinder, P. Perona, and S. Belongie. Visual recognition with humans in the loop. In *European Conference on Computer Vision*, 2010.

W. Croft and R. Das. Experiments with query acquisition and use in document retrieval systems. In *Proceedings of the 13th International Conference on Research and Development in Information Retrieval*, pages 349–368, 1990.

S. Dasgupta, A. Dey, N. Roberts, and S. Sabato. Learning from discriminative feature feedback. In *Advances in Neural Information Processing Systems*, pages 3955–3963, 2018.

G. Druck, G. Mann, and A. McCallum. Learning from labeled features using generalized expectation criteria. In *Proceedings of ACM Special Interest Group on Information Retrieval*, 2008.

A. Y. Kogan. Disjunctive normal forms of boolean functions with a small number of zeros. *USSR Computational Mathematics and Mathematical Physics*, 27(3):185–190, 1987.

O. Mac Aodha, S. Su, Y. Chen, P. Perona, and Y. Yue. Teaching categories to human learners with visual explanations. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

Y. V. Maximov. Implementation of boolean functions with a bounded number of zeros by disjunctive normal forms. *Computational Mathematics and Mathematical Physics*, 53(9):1391–1409, 2013.

D. Mubayi, G. Turán, and Y. Zhao. The dnf exception problem. *Theoretical computer science*, 352(1-3): 85–96, 2006.

S. Poulis and S. Dasgupta. Learning with feature feedback. In *Twentieth International Conference on Artificial Intelligence and Statistics*, 2017.

H. Raghavan, O. Madani, and R. Jones. Interactive feature selection. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, pages 841–846, 2005.

B. Settles. Closing the loop: fast, interactive semi-supervised annotation with queries on features and instances. In *Empirical Methods in Natural Language Processing*, 2011.

R. Visotsky, Y. Atzmon, and G. Chechik. Learning with per-sample side information. In *AGI*, 2019.

Y. I. Zhuravlev. Realization of boolean functions with a small number of zeros by disjunctive normal forms and related problems. *Soviet Mathematics-Doklady*, 32(3):771–775, 1985.

J. Zou, K. Chaudhuri, and A. T. Kalai. Crowdsourcing feature discovery via adaptively chosen comparisons. In *Conference on Human Computation and Crowdsourcing (HCOMP)*, 2015.