Learning from discriminative feature feedback

Sanjoy Dasgupta, Akansha Dey, Nicholas Roberts
Department of Computer Science and Engineering
University of California, San Diego
dasgupta@eng.ucsd.edu,n3robert@ucsd.edu,a1dey@ucsd.edu

Sivan Sabato
Department of Computer Science
Ben-Gurion University of the Negev
sabatos@cs.bgu.ac.il

Abstract

We consider the problem of learning a multi-class classifier from labels as well as simple explanations that we call *discriminative features*. We show that such explanations can be provided whenever the target concept is a decision tree, or can be expressed as a particular type of multi-class DNF formula. We present an efficient online algorithm for learning from such feedback and we give tight bounds on the number of mistakes made during the learning process. These bounds depend only on the representation size of the target concept and not on the overall number of available features, which could be infinite. We also demonstrate the learning procedure experimentally.

1 Introduction

Communication between humans and machine learning systems has typically been restricted to labels alone. A human provides labels for a data set and in return gets a classifier that predicts labels of new instances. This is a lot more rigid than the way humans learn. The field of *interactive learning* explores richer learning frameworks in which the machine engages with the human (or other sources of annotation) while learning is taking place, and the communication between the two is allowed to be more varied. A key question in this enterprise is whether such interaction can overcome fundamental algorithmic and statistical barriers to learning.

The one interactive framework that has perhaps been explored the most thoroughly is *active learning of classifiers*. Here, the learning machine begins with just a pool of unlabeled data, and interacts with a human by asking for labels of specific points. By adaptively focusing on maximally informative points, it can sometimes dramatically reduce the amount of labeling effort. Indeed, over the past two decades, a number of active learning algorithms have been designed that provably require only logarithmically as many labels as random querying, or otherwise reduce the labeling requirement considerably, in a variety of canonical settings, e.g. [1, 7].

In this paper, our interest is in feedback that goes beyond labels, to *simple explanations* of a particular type. Imagine, for instance, that you have just finished watching a movie in your living room, and your electronic assistant—Siri, Alexa, or one of their colleagues—asks "Did you like the movie?" You dutifully reply "yes", in effect providing a labeled data point. But then you casually add "I really like John Hurt". This last piece of information is spontaneous and does not require extra effort. But it can be far more useful, for the purposes of determining what movie to recommend next, than a mere thumbs-up or thumbs-down label. This kind of explanation could be called *feature feedback*, because it helps identify relevant features in a high-dimensional space of rationales for user preferences.

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

Feedback of this form has been used effectively in information retrieval [4, 13, 6] and in computer vision [10, 9, 8]. In general, however, these systems have been geared towards specific applications, and it is of interest to study such feedback rigorously, in a more abstract setting.

1.1 Predictive versus discriminative feature feedback

There has been a significant amount of work on what might be called *predictive feature feedback*. Suppose, for example, that in a document classification task, a labeler assigns each document x to a category y ("sports", "politics", and so on). While making this determination, the labeler might also be able to highlight a few words that are highly indicative of the label (e.g. "Congress", "filibuster"). This kind of auxiliary feedback has been explored in a variety of empirical studies, for text and image, with promising results [4, 13, 6, 14, 8]. More recently, the theoretical results of [12] have shown that such feedback can improve the rate of convergence of learning.

In this paper, we study an alternative setting that we call *discriminative feature feedback*. Consider a computer vision system that is learning to recognize different animals. Whenever it makes a mistake—classifies a "zebra" as a "horse", say—a human labeler ("the teacher") corrects it. While doing this, the labeler can also, at little extra cost, highlight a part of the image (the stripes, for instance) that distinguishes the two animals. Work on recognizing different species of birds, for instance, has used this sort of feedback effectively [3].

This kind of discriminative feedback is quite different from the predictive feature feedback of earlier work. In the document example, the feedback yields *predictive features*: the presence of the word "filibuster" is a moderately-strong predictor of the label "politics". In contrast, discriminative features are not necessarily predictive for the entire class. In the animal example above, "stripes" are not predictive of the class of zebras, since many animals have stripes. But they do distinguish zebras from horses. Thus discriminative feedback can be advantageous in a multi-class setting: the feature need only differentiate between two classes, rather than separating one entire class from all the others. In our abstract model, we relax this even further by requiring only that a discriminative feature separate one *subcategory* of a certain label from a subcategory of a different label.

1.2 Contributions

Our first contribution is to formalize one particular type of discriminative feature feedback. Human explanations, even simple ones, are rife with ambiguity and thus it is important to design explanatory mechanisms that have precise semantics.

Next, we present a simple and efficient learning algorithm that uses this feedback for multi-class classification. It operates in the online learning framework: at each point in time, a new data point arrives, it is classified by the learner's current model, and then feedback is received if this classification is incorrect. We show that the algorithm provably converges to the correct concept, and we provide a tight mistake bound on the total number of errors it makes over the lifetime of the learning process.

What concept classes are learned by our algorithm? We show that it can efficiently learn any multi-class concept that can be expressed as a decision tree, with a mistake bound that is quadratic in the number of leaves of the tree. More generally, it works when the target concept is expressible using a particular multi-class version of DNF (disjunctive normal form, OR-of-ANDs) formulas that we call *separable DNF*. In the setting of binary classification, disjunctive normal form concepts have proved to be computationally intractable to learn under standard supervised learning [11, 5], which is a bit troubling since humans seem to use such concepts quite naturally. But the hardness results apply to situations where the only feedback on any example is its label. With richer feedback, learning becomes easy and efficient.

The model learned by the algorithm is a logical combination of features obtained during feedback, a sort of decision list where each individual entry in the list is a conjunction. The mistake bound has no dependence on the overall number of available features, which could potentially be infinite. As a result this methodology can be used to build classifiers based on vast pools of low-level named features, which neural nets are increasingly able to provide.

Lastly, we demonstrate the learning procedure experimentally.

1.3 Other related work

Related to our work is the *infinite attribute model* of Blum [2], which introduces an online learning framework in which the goal is to learn a standard classifier, such as a linear separator, in situations where the total number of available features is infinite. Earlier mistake bounds for online learning, such as those for the Perceptron and Winnow algorithms, had some dependence on the number of features. It was shown in [2] that this dependence can be removed if, for any given example x, only finitely many features are present. These can, for instance, be thought of as the features that are perceptually most salient. In our paper, we again consider an online learning scenario and give a mistake bound that has no dependence on the overall number of available features. In our case, this arises naturally as a by-product of constructing classifiers from feature feedback, which is a different mechanism technically but does ultimately connect back to perceptual salience.

A different type of feedback was studied in [15], for the specific purpose of learning DNF formulas. The learner is allowed to make queries in which it provides two data points from the same class and asks how many terms in the DNF formula they both satisfy.

2 The feedback model

We introduce our formal model by means of an example. Let's say the goal is to learn a classifier that takes as an input example a description of an animal—given by a set of features describing where it lives, what it eats, its appearance, and so on—and then classifies it as `mammal`, `bird`, `reptile`, `amphibian`, or `fish`.

The learning takes place in rounds of interaction. On each round,

- A new animal is presented, e.g.,

  seahorse
- The learner classifies this instance using its current model.
  - E.g., it (mistakenly) says:

    $\gg$ `mammal`
  - In addition, the learner provides an example of an animal it has seen previously, that it considers to be similar to the new animal, and that belongs to the predicted class. E.g.,

    $\gg$ *seahorse* is similar to *horse*, which is a `mammal`.
- The labeler responds if the prediction is incorrect.
  - The labeler provides the correct class.

    $>$ Correct class: `fish`
  - In addition, the labeler provides a distinguishing feature between the instance being classified and the instance that the learner suggested as a similar example.

    $>$ Distinguishing feature between *seahorse* and *horse*: **lives-in-water**

Here the feature, **lives-in-water**, doesn't distinguish all fish from all mammals—some mammals do live in water, for instance—but does distinguish a group of fish that includes seahorses from a group of mammals that include horses.

We now formalize the semantics of this kind of interaction.

2.1 An abstract model of interaction

Let c^* be the target concept to be learned, where c^* is a mapping from the input space $\mathcal{X}$ (e.g., animals) to a finite label space $\mathcal{Y}$ (e.g., $\{\texttt{mammal}, \texttt{bird}, \texttt{reptile}, \texttt{amphibian}, \texttt{fish}\}$). The learner has access to a set of Boolean features Φ on $\mathcal{X}$, and expresses concepts in terms of these. For instance, in the example above, one of the features $\phi \in \Phi$ is **lives-in-water**.

Informally, the concepts we can handle are those in which: the data has some unknown underlying clusters (e.g., "regular land mammals", "egg-laying mammals", "marine mammals", "bony fish", "cartilaginous fish", etc); each label corresponds to a union of some of these clusters; and any two clusters with different labels can be separated by a single feature.

- A new instance x_t arrives.
- The learner supplies:
  - A prediction $\widehat{y}_t$
  - A previously-seen instance $\widehat{x}_t$ with that label

  E.g.: "I think it is a `mammal`; it is similar to a *horse*"
- If the prediction is correct: no feedback is obtained.
- If the prediction is incorrect:
  - The teacher provides the correct label y_t
  - The teacher also provides a feature ϕ that separates $G(x_t)$ from $G(\widehat{x}_t)$:

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

  E.g.: "No, it is a `fish`: **lives-in-water**"

Figure 1: Round t of interaction.

Formally, we assume that $\mathcal{X}$ can be represented as the union of m sets, $\mathcal{X} = G_1 \cup G_2 \cup \cdots \cup G_m$, possibly overlapping. This representation (that is, the identity of the sets in the union) is unknown to the learner, and satisfies the following:

- Each of the sets is pure in its label: for each i, there exists a label $\ell(G_i) \in \mathcal{Y}$ such that

  $$c^*(x) = \ell(G_i) \quad \text{for all } x \in G_i.$$

  (It follows that two sets G_i, G_j can intersect only if they have the same label.)
- Any two sets G_i, G_j with $\ell(G_i) \neq \ell(G_j)$ have a *discriminating feature*: there is some $\phi \in \Phi$ and $b \in \{0, 1\}$ such that

  $$\phi(x) = \begin{cases} b & \text{if } x \in G_i, \\ \neg b & \text{if } x \in G_j. \end{cases}$$

  For instance, if G_i is the cluster of land mammals and G_j is the cluster of cartilaginous fish, then one possible separating feature is **lives-in-water**.

As we discuss below, a representation that satisfies the assumptions above naturally exists if the set of Boolean features Φ is sufficiently rich. We place no restrictions on the number of features, which can even be infinite. Moreover, since the algorithm and the mistake bound have no dependence on the number of features in Φ, the requirement that a *single* feature be used to separate sets is not restrictive: one can always include in Φ "single features" that are combinations of other basic features.

2.2 The learning protocol

Figure 1 shows how the tth round of interaction proceeds. In the protocol, $G(x)$ is the set (one of $G_1, \ldots, G_m$) containing x. If there are multiple such sets, it is some particular choice. Thus $G : \mathcal{X} \to \{G_1, \ldots, G_m\}$ and $x \in G(x)$.

To reiterate, the labeler does not need to provide features that separate entire classes from one another. Rather, these features just need to separate *a subgroup of one class*, containing x_t, from a subgroup of another class, containing $\widehat{x}_t$. These subgroups might just be the singletons x_t and $\widehat{x}_t$, in which case the labeler need only distinguish these two specific instances. But it is reasonable to expect that a labeler will attempt to find features that are fairly general, in which case these subgroups will be somewhat larger. The clusters in the feedback formalism reflect the *level of categorization* at which the labeler is operating. They are allowed to be of arbitrary size; however, the complexity of the algorithm we later present will depend upon the total *number* of clusters. If there are m clusters, the total number of mistakes made by the algorithm will be bounded by m^2.

2.3 When do the assumptions hold?

Our assumptions posit that there is some representation of the domain as purely-labeled sets that can be separated by single features, and that this representation is the one used implicitly by the teacher when providing feedback to the learner. An important question is when these assumptions hold. Moreover, since the number of sets in the representation determines the mistake bound of our learner, we would like to identify when this number is small.

First, note that whenever the set of features Φ is sufficiently rich that any two instances x, x' differ on at least one feature, we can always have a representation based on singleton sets which satisfies the assumptions. In this sense, the model is non-parametric, and allows *any* target concept. However, the complexity of learning depends upon the number of sets, which we are denoting by m, and would be trivial if $m = |\mathcal{X}|$. Therefore, we would like to identify concepts which admit $m \ll |\mathcal{X}|$.

One case in which this holds is when the target concept c^* can be expressed as a decision tree with m leaves, using features in Φ. We can define the set of examples $x \in \mathcal{X}$ that reach the j-th leaf to be a cluster G_j, so that there are m clusters. This way, any two clusters G_i and G_j can be separated by a single feature: the feature ϕ at the internal node that is the lowest common ancestor of leaves i and j.

More generally, discriminative feedback with m clusters is possible if and only if the target concept is expressible as a particular kind of multi-class disjunctive normal form formula over the features in Φ, that we call *separable-DNF*, and that has at most m terms.

Definition 1 (Separable-DNF concept) *A separable-DNF concept over features Φ is a concept $c^* : \mathcal{X} \to \mathcal{Y}$ such that each individual class $y \in \mathcal{Y}$ is characterized by a DNF formula F_y, where x satisfies F_y if and only if $c^*(x) = y$, and:*

- *The literals in each F_y are individual features from Φ or negations of such features.*
- *For any $y \neq y'$, denote $F_y = F_{y,1} \vee F_{y,2} \vee \cdots$ and $F_{y'} = F_{y',1} \vee F_{y',2} \vee \cdots$, where the $F_{y,i}$ and $F_{y',i}$ are conjunctions. Then for any $F_{y,i}$ and $F_{y',j}$, there is some feature $\phi \in \Phi$ that is in both conjunctions, but with opposite polarity.*

We say that the separable-DNF concept is of size m if the total number of conjunctions $F_{y,i}$ is m.

Lemma 1 *Target concept c^* can be represented using $G_1, \ldots, G_m$ which satisfy the assumptions in Section 2.1 if and only if c^* is a separable-DNF concept of size m over features Φ.*

The proof is deferred to Appendix A.

2.4 DNF formulas for binary classification

There is a large body of work on learning disjunctive normal form formulas for *binary* classification. These concepts don't exactly fall into the framework above because they are asymmetric: there is a DNF formula for positive instances, and everything else is a negative instance. In Appendix B, we provide a simple variant of our learning algorithm specifically for this case. For a target DNF formula with m terms, each containing k literals, it makes at most $m(k+1)$ mistakes.

3 A mistake-bounded learning algorithm

We now show that under the setting of discriminative feature feedback, there is an efficient learning algorithm that reaches the target concept c^* after making at most m^2 mistakes.

3.1 The algorithm

The algorithm is shown in Figure 2. It maintains:

- a list L of some of the instances seen so far;
- for each item x in this list, its label as well as a conjunction $C[x]$ that holds true for all of the cluster $G(x)$; and
- a default instance and label (x_o, y_o) to apply to examples that violate all conjunctions of L.

```
Initialization:
    • Get the label $y_o$ of the first example $x_o$ (these serve as a default prediction)
    • Initialize $L$ to an empty list

At time $t$, given a new point $x_t$:
    • If there exists $\widehat{x} \in L$ such that $x_t$ satisfies $C[\widehat{x}]$:
        – Predict label$[\widehat{x}]$ and provide example $\widehat{x}$
        – If incorrect:
            ∗ Get correct label $y_t$ and feature $\phi$
            ∗ $C[\widehat{x}] := C[\widehat{x}] \wedge \neg\phi$
    • Else:
        – Predict default label $y_o$ and provide example $x_o$
        – If incorrect:
            ∗ Get correct label $y_t$ and feature $\phi$
            ∗ Add $x_t$ to $L$
            ∗ Set label$[x_t] := y_t$ and $C[x_t] := \phi$
```

Figure 2: An algorithm that learns from discriminative feature feedback.

The conjunctions $C[x]$ are built entirely out of features obtained from the teacher. We denote by $\mathcal{C}_L = \{C[x] : x \in L\}$ the set of conjunctions for the examples in L.

3.2 Mistake bounds

Theorem 2 *Suppose that the algorithm of Figure 2 is used to learn a target concept c^*, and the teacher provides discriminative feature feedback corresponding to a representation $G_1, \ldots, G_m$ which satisfies the assumptions in Section 2.1. Then the total number of mistakes made over the lifetime of the algorithm is at most m^2.*

We prove this in several steps. First, we note two key invariants of the algorithm:

(I-1) Any item $x \in L$ has been seen in a previous round.

(I-2) For any $x \in L$, label$[x]$ is correct and every point in $G(x)$ satisfies $C[x]$.

The first invariant is trivial. The second invariant holds since the label is provided by the teacher, and the conjunction $C[x]$ is a conjunction of literals taken from the teacher. These literals are all satisfied by $G(x)$, as defined in the learning protocol in Figure 1.

Next, we show that L contains at most one representative per group G_i.

Lemma 3 *For any distinct $x, x' \in L$, we have $G(x) \neq G(x')$.*

PROOF: The only time a new x is added to L is in situations when it doesn't satisfy any of the conjunctions in L, and is therefore not in any of the corresponding $G(\cdot)$, as per invariant (I-2). □

The above observations allow deriving a connection between the size of the conjunctions in $\mathcal{C}_L$ and the number of mistakes the algorithm makes.

Lemma 4 *Let B be an upper bound on the total number of literals in any conjunction in the list. Then the number of mistakes made by the algorithm is at most mB.*

PROOF: On each mistake of the algorithm, either: (i) an existing $x \in L$ has its conjunction restricted by an additional literal or (ii) a new item x is added to L with a conjunction of size 1. Thus, the total number of literals in conjunctions in L is equal to the number of mistakes made by the algorithm. By assumption, each conjunction in L has $\leq B$ literals, and by Lemma 3 there are at most m such conjunctions. □

We can now prove the mistake bound stated above.

Proof of Theorem 2: We show that each conjunction in L has at most m literals as follows. First, any conjunction $C[\widehat{x}]$ always starts with a single literal. Subsequently, the conjunction is extended only when some instance x_t appears that satisfies the conjunction and yet has $G(x_t) \neq G(\widehat{x})$. In this case, one literal is added to $C[\widehat{x}]$ and thereafter no instance in $G(x_t)$ satisfies the conjunction. Since there are m different sets G_i, it follows that there can be at most $m-1$ rounds in which $C[\widehat{x}]$ is extended. Thus $C[\widehat{x}]$ has at most m literals. The mistake bound now follows from Lemma 4. □

Further, we show that this mistake bound is nearly tight, as stated in the following theorem and proved in Appendix C.

Theorem 5 *The worst-case mistake bound of the algorithm in Figure 2, assuming discriminative feature feedback with m clusters, is at least $(m-1)(m-2)/2$.*

It is also possible to derive a mistake bound which depends on the total number of features that the teacher uses during the interaction. This bound is useful if the teacher uses a small number of different features to discriminate between clusters. This can happen if the teacher attempts to reuse features, or if the target conjunction is sparse.

Theorem 6 *Under the same assumptions as in Theorem 2, if the teacher uses at most k features during the running of the algorithm, then the total number of mistakes made over the lifetime of the algorithm is at most km.*

PROOF: If the teacher uses at most k different features during the run of the algorithm, then each conjunction in L uses at most k literals: Due to invariant (I-2), each conjunction is satisfied by at least one instance, thus it cannot include both a feature and its negation. The mistake bound follows from Lemma 4. □

4 An illustrative experiment

The ZOO data set from the UCI ML repository contains information on 101 animals: for each, one of seven labels (`mammal`, `bird`, `reptile`, `fish`, `amphibian`, `insect`, `other`) as well as 21 Boolean features. The goal is to learn a classifier that predicts the label from the 21 features.

The learning algorithm of Figure 2 can potentially return different classifiers depending on the particular ordering of the data points. Figure 3 shows the result of one run. The starting example, and thus default prediction, is *frog* (class `amphibian`), and a total of 14 mistakes are made before convergence. On each mistake, a separating feature is chosen *at random* from those that distinguish the two instances (x_t and $\widehat{x}$ in the notation of Figure 2). A human teacher would perhaps make more judicious feature choices.

The first table in the figure shows the prediction and similar example provided by the learner (columns 2 and 3) in each mistake round, as well as the correct label and discriminative feature provided by the labeler (columns 4 and 5). The final decision list L is shown below. Recall that an instance is classified by first going through all the conjunctions in L and then falling through to the default prediction if there is no match.

Notice that the list L contains exactly 14 literals in all, one from each mistake. Also, we know from Lemma 3 that each of the "underlying groups" contributes at most one rule in L. Therefore, the method of choosing features implicitly divides the class `reptile` into at least two groups, which appear to correspond to land reptiles and aquatic reptiles, and divides the somewhat nebulous class `other` into two groups, exemplified by *worm* and *lobster*.

5 Conclusions and further directions

There is potential for enhancing the scope, robustness, and ease-of-use of learning systems by having them learn from simple explanations, and in turn explain their predictions. A crucial part of this enterprise is to identify and formalize simple explanatory mechanisms, and to study how they can be

Instance	Prediction	Similar instance	True label	Discriminating feature
frog			amphibian	(default prediction)
worm	amphibian	frog	other	not(backbone)
girl	amphibian	frog	mammal	milk
herring	amphibian	frog	fish	zero-legs
seasnake	fish	herring	reptile	not(fins)
hawk	amphibian	frog	bird	two-legs
tortoise	amphibian	frog	reptile	not(aquatic)
termite	other	worm	insect	six-legs
lobster	amphibian	frog	other	not(backbone)
ladybird	reptile	tortoise	insect	six-legs
honeybee	other	lobster	insect	airborne
housefly	amphibian	frog	insect	not(aquatic)
flea	other	lobster	insect	breathes
seasnake	amphibian	frog	reptile	tail
newt	reptile	seasnake	amphibian	not(zero-legs)

Final decision list (L):

- not(backbone) AND not(six-legs) ⟹ other (*worm*)
- milk ⟹ mammal (*girl*)
- zero-legs AND fins ⟹ fish (*herring*)
- two-legs ⟹ bird (*hawk*)
- not(aquatic) AND not(six-legs) ⟹ reptile (*tortoise*)
- not(backbone) AND not(airborne) AND not(breathes) ⟹ other (*lobster*)
- not(aquatic) ⟹ insect (*housefly*)
- tail AND zero-legs ⟹ reptile (*sea snake*)
- ELSE: amphibian (*frog*)

Figure 3: The 14 mistakes made on the ZOO data and the final classifier.

handled algorithmically. This paper introduces a novel type of explanation that is fairly intuitive, and shows that it lends itself to simple learning algorithms with rigorous performance guarantees. It is a first step, and a wide range of open problems remain.

For this particular feedback scheme, what happens if the labeler is noisy or careless? We know from Section 2 that the scheme can in principle handle any concept, and thus noisy feedback is unlikely to derail convergence: if necessary, for instance, a noisily-labeled point can be in a cluster of its own. A more subtle issue is how much the mistake bound can blow up as a result of noise.

The decision lists produced by the algorithm of Figure 2 are accompanied by reassuring guarantees. However, is it possible to retain those guarantees while producing a model that is more concise? This open problem is particularly important in cases where the lists are intended to be interpretable.

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 SD and SS were at the “Foundations of Machine Learning” program at the Simons Institute for the Theory of Computing, Berkeley.

References

[1] M.-F. Balcan, A. Beygelzimer, and J. Langford. Agnostic active learning. In *Proceedings of the 23rd International Conference on Machine Learning*, 2006.

[2] A. Blum. Learning boolean functions in an infinite attribute space. *Machine Learning*, 9(4):373–386, 1992.

[3] 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.

[4] W.B. 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.

[5] A. Daniely and S. Shalev-Shwartz. Complexity theoretic limitations on learning dnfs. In *Conference on Learning Theory*, pages 815–830, 2016.

[6] 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.

[7] S. Hanneke. Rates of convergence in active learning. *Annals of Statistics*, 39(1):333–361, 2011.

[8] 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.

[9] D. Parikh and K. Grauman. Relative attributes. In *Proceedings of the International Conference on Computer Vision*, 2011.

[10] P. Perona. Visions of a Visipedia. *Proceedings of the IEEE*, 98(8):1526–1534, 2010.

[11] L. Pitt and L.G. Valiant. Computational limitations on learning from examples. *Journal of the ACM*, 35(4):965–984, 1988.

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

[13] 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.

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

[15] L. Yang, A. Blum, and J. Carbonell. Learnability of DNF with representation-specific queries. In *Innovations in Theoretical Computer Science*, 2013.

A Proof of Lemma 1

We prove the two parts of Lemma 1 via the following two lemmas.

Lemma 7 *Suppose the target concept c^* can be represented using $G_1, \ldots, G_m$ which satisfy the assumptions in Section 2.1. Then c^* is a separable-DNF concept of size m over features Φ.*

PROOF: For any two clusters G_i and G_j with different labels, there is some feature $\phi_{ij} \in \Phi$ that separates the two. For any G_i, let conj(G_i) be a conjunction of the features that separate it from sets with different labels: $\{\phi_{ij} : 1 \leq j \leq m$ such that $\ell(G_i) \neq \ell(G_j)\}$, each with the correct polarity.

Now, for any label $y \in \mathcal{Y}$, we can define the DNF formula

$$F_y = \bigvee_{i:\ell(G_i)=y} \text{conj}(G_i).$$

The formulas $\{F_y : y \in \mathcal{Y}\}$ together constitute a separable-DNF expression for the target concept. □

The converse is also true, as given by the following lemma.

Lemma 8 *Suppose target concept c^* is a separable-DNF concept of size m over features Φ. Then c^* can be represented using $G_1, \ldots, G_m$ which satisfy the assumptions in Section 2.1.*

PROOF: By assumption, for each label $y \in \mathcal{Y}$, there is a DNF formula $F_y = F_{y,1} \vee F_{y,2} \vee \cdots$. Define clusters $G_{y,i} = \{x \in \mathcal{X} : x$ satisfies $F_{y,i}\}$. There are m clusters $G_{y,i}$ in total, each is pure in its label, and it follows from the definition of separable-DNF that any two clusters with different labels have a separating feature. □

B Binary classification with DNF formulas

The algorithm of Figure 2 produces a classifier consisting of a list of conjunctions, each indexed by a representative example. In effect, there is a disjunctive normal form formula for every class.

The extensive literature on learning DNF has adopted a slightly different formulation: it has focused on binary classification, where there is a DNF formula for the positive instances, while the negative instances consist of the complement. In this section, we give a variant of our framework that specifically produces models of this type. Our aim is to make it clear that learning DNF, in its usual formulation, is easy with discriminative feature feedback.

B.1 Feedback model

Let $\mathcal{X}$ denote the instance space. We consider a setting where the target classifier labels each point in $\mathcal{X}$ as either positive or negative. On round t of interaction:

- A new instance x_t is presented.
- Either the learner labels it "$-$":
  - The teacher supplies the correct label.
- Or the learner labels it "$+$":
  - In this case, the learner additionally provides an instance $\widehat{x}$ that it has seen before, that it knows to be "$+$", and that it considers similar.
  - The teacher supplies the correct label.
  - If the correct label is "$-$": the teacher supplies a feature $\phi \in \Phi$, with positive or negative polarity, that is true of $\widehat{x}$ but not of x_t.

We now formalize the semantics of the feedback. Let $\mathcal{X}^+$ denote the set of positive instances and $\mathcal{X}^-$ the negative instances. We will assume that the teacher has in mind (perhaps implicitly) some grouping $\mathcal{X}^+ = G_1 \cup G_2 \cup \cdots \cup G_m$, where the G_i are possibly overlapping, such that:

- The teacher associates instance $x \in \mathcal{X}^+$ with group $G(x) \in \{G_1, \ldots, G_m\}$.

```
Initialization:
    • Initialize $L$ to an empty list

At time $t$, given a new point $x_t$:
    • If there exists $\widehat{x} \in L$ such that $x_t$ satisfies $C[\widehat{x}]$:
        – Predict label "$+$" and provide example $\widehat{x}$
        – If incorrect:
            ∗ Teacher gives label "$-$" and feature $\phi$
            ∗ $C[\widehat{x}] := C[\widehat{x}] \wedge \phi$
    • Else:
        – Predict label "$-$"
        – If incorrect:
            ∗ Teacher gives label "$+$"
            ∗ Add $x_t$ to $L$
            ∗ Set $C[x_t] := \emptyset$ (by convention, all instances satisfy the empty conjunction)
```

Figure 4: An algorithm that learns a binary DNF classifier from discriminative feature feedback.

- For any $1 \leq i \leq m$, there is a set of k features $\phi_{i,1}, \ldots, \phi_{i,k} \in \Phi$ (with either positive or negative polarity) such that the conjunction $\phi_{i,1} \wedge \cdots \wedge \phi_{i,k}$ is satisfied by all instances in G_i and by no instances in $\mathcal{X}^-$.
- When the teacher needs to separate an instance $\widehat{x} \in \mathcal{X}^+$ from some $x' \in \mathcal{X}^-$, where $G(\widehat{x}) = G_i$, he/she chooses one of the features $\phi_{i,j}$.

It is clear that such feedback is possible if and only if the positive instances $\mathcal{X}^+$ can be represented by a DNF formula in which there are m terms, each with up to k literals drawn from Φ.

B.2 Algorithm

The algorithm is shown in Figure 4. It maintains a list L consisting of some of the positive instances it has seen. For each $\widehat{x}$ in this list, it has a conjunction $C[\widehat{x}]$ that starts off empty and grows by one literal every time the algorithm makes a mistake in which it provides $\widehat{x}$ as the similar instance.

Throughout the algorithm, the following invariants are maintained.

(I-1) Every x in list L is a positive instance.

(I-2) For every x in the list, the conjunction $C[x]$ is satisfied by all of $G(x)$.

These hold by design, and by the assumption on the teacher's feedback.

B.3 Mistake bound

The analysis of the algorithm follows that of the multi-class version. The first step is to establish that the entries of L correspond to distinct clusters G_i; the argument is exactly as before.

Lemma 9 *For any distinct x, x' in L, we have $G(x) \neq G(x')$.*

Lemma 10 *The number of mistakes made by the learner is at most $m(k+1)$.*

PROOF: Each time a mistake occurs, either (1) an existing conjunction is augmented by one literal, or (2) a new entry is added to L. In light of Lemma 9, the second option can occur at most m times. Once a new element x has been added to L, the conjunction $C[x]$ can grow at most k times by the assumption on the teacher's feedback: if $G(x) = G_i$, say, then any feedback on x is limited to the features $\phi_{i,1}, \ldots, \phi_{i,k}$. □

C Proof of Theorem 5

We prove below Theorem 5, which states that the worst-case mistake bound of our algorithm is at least $(m-1)(m-2)/2$, where m is the number of clusters defined in Section 2.1.

PROOF: Let $\mathcal{X} = \{0,1\}^d$, and let Φ be the list of literals based on single coordinate values, denoted as (i,b) for $i \in \{1,\ldots,d\}, b \in \{0,1\}$, where the feature (i,b) is true if $x_i = b$. Let the labels be $\mathcal{Y} = \{1,\ldots,d,d+1\}$. Consider the concept c^* where the label of x is the index of the smallest coordinate in the vector x with a value of 1, or $d+1$ if it is the all-zero vector. Let $m = d+1$. Let the representation $G_1,\ldots,G_m$ be such that $G_i = \{x \in \mathcal{X} : c^*(x) = i\}$. Note that for each $i < j$, coordinate i discriminates between G_i, G_j: it is 1 in G_i and 0 in G_j.

Let M be the maximal number of mistakes made by the algorithm on any sequence of interactions with $\mathcal{X}$, and suppose for contradiction that $M < (m-1)(m-2)/2$. Consider a run in which this number is reached, and the state of the algorithm after this number of mistakes have been made. We consider possible cases and show that in each case there is some instance that, if presented at this point in time to the algorithm, can result in a mistake, contradicting the maximality of M. Denote $L_+ = L \cup \{x_o\}$, where (x_o, y_o) is the default prediction used by the algorithm.

- If there is some label that has no representative in L_+, then any instance with that label will be wrongly classified.
- Otherwise, every label has at least one representative in L_+. Therefore, at least $m-1$ labels have a representative in L. As argued in the proof of Lemma 4, the total number of literals in $\mathcal{C}_L$ is equal to the number of mistakes made by the algorithm. In our case, the total number of literals in conjunctions in $\mathcal{C}_L$ is $M < (m-1)(m-2)/2 = \sum_{i=1}^{m-1}(i-1)$. Therefore, there is at least one label i such that for $x \in L$ with label i, its conjunction $C[x]$ has less than $i-1$ literals. Therefore, at least one coordinate j out of $1,\ldots,i-1$ is not constrained by $C[x]$. Note that coordinates up to $i-1$ that *are* constrained by $C[x]$ must be constrained to zero by the definition of the category i. Let z be an example with label j such that its first $j-1$ coordinates are zero, $z_j = 1$, and the other coordinates are consistent with $C[x]$. Then the algorithm wrongly labels this example as i.

We have reached a contradiction in all cases, which proves that the mistake bound must be larger than M. □