Understanding Classifier Errors by Examining Influential Neighbors

Mayank Kabra, Alice Robie, Kristin Branson
Janelia Research Campus of the Howard Hughes Medical Institute
Ashburn, VA, 20147, USA
{kabram,robiea,bransonk}@janelia.hhmi.org

Abstract

Modern supervised learning algorithms can learn very accurate and complex discriminating functions. But when these classifiers fail, this complexity can also be a drawback because there is no easy, intuitive way to diagnose why they are failing and remedy the problem. This important question has received little attention. To address this problem, we propose a novel method to analyze and understand a classifier's errors. Our method centers around a measure of how much influence a training example has on the classifier's prediction for a test example. To understand why a classifier is mispredicting the label of a given test example, the user can find and review the most influential training examples that caused this misprediction, allowing them to focus their attention on relevant areas of the data space. This will aid the user in determining if and how the training data is inconsistently labeled or lacking in diversity, or if the feature representation is insufficient. As computing the influence of each training example is computationally impractical, we propose a novel distance metric to approximate influence for boosting classifiers that is fast enough to be used interactively. We also show several novel use paradigms of our distance metric. Through experiments, we show that it can be used to find incorrectly or inconsistently labeled training examples, to find specific areas of the data space that need more training data, and to gain insight into which features are missing from the current representation.

1. Introduction

Classifiers produced by modern machine learning algorithms such as boosting [14, 15] and support vector machines [6] are extremely complex, often with thousands of variables. Developing systems that depend on these opaque classifiers can be frustrating, because when a classifier makes mistakes, the engineer has little information as to why the classifier made the mistakes and how to improve its performance. Currently, the recourses available are to increase the training data set size or to examine the errors and guess what types of training data or features might be missing. This problem is growing in importance as systems allowing lay-users to train their own classifiers are emerging [27, 19, 20]. Even for machine learning experts, it is nearly impossible to understand, navigate, and visualize large, high-dimensional data sets [16, 23, 8, 10].

Our proposed method to gain an understanding of a classifier's errors for a given data set hinges on the insight that its prediction on an example is not equally influenced by all examples in the training set. This is obvious for classifiers such as nearest neighbor, but is also true for more complex classifiers like boosting. But unlike nearest neighbor, which examples are the most influential for boosting depends on the classification task. For example, the influence of a training example will not depend on features that are not relevant in separating the two classes. We propose that the influence of a training example x' on a test example x can be found by training two classifiers: one including $(x', +1)$ and the other including $(x', -1)$. The training example that most changes the prediction for the given test example is the most influential (Figure 1(a)).

Thus, to understand why a particular test example is mispredicted, the few training examples that are most influential can be selected for analysis. By viewing these examples, the engineer can focus on and visualize the small portions of the data space relevant to that error. This will allow the engineer to better understand the cause of the failure. For example, it is difficult to deduce why an object recognition system might label ceiling fans as chairs (Figure 1(b)). But, if we knew that the most influential examples in the training set are of the chairs shown, then it is easy to realize that the classifier was confused by the fan-shaped base of the chair.

However, for the majority of classifier families, finding which training examples are most influential is computationally impractical because it requires training a new classifier for each training example. To make computation of influence fast enough to be used in interactive, real-time systems, we propose a novel dissimilarity metric that approximates influence. We borrow the idea of *version space*,

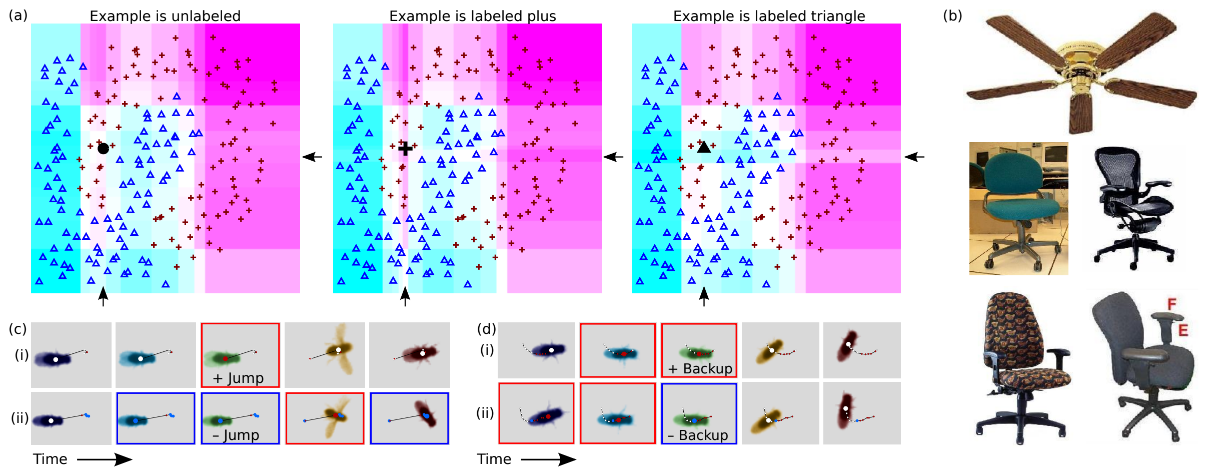


Figure 1. (a) Illustration of the influence of an example. Two classes of training data are indicated by + (red pluses) and △ (blue triangles). If we add the example indicated by the black circle to the training data set with the plus and triangle labels, the predictions of the classifiers learned using boosting (indicated by the color and intensity of the image) change in certain parts of the space. In particular, we see a magenta column when we add the example as red plus, and a cyan row when we add the example as a blue triangle. The effect of changing the example's label is a function of both the learning algorithm (boosted decision stumps) and the distribution of the data. (b) Images of a ceiling fan and chairs from the Caltech 101 data set [12] that look similar to an object detector. If an object detector predicted this fan was a chair, without the context that some chairs have fan-shaped bases, it would be difficult to understand the cause of this error. (c) Example of a labeling mistake in a data set used for training a "jump" behavior classifier for flies. (d) Example of inconsistently labeled data in a training set used to train a "backup" behavior classifier for flies. The central frames are virtually identical, but were given opposite labels by the user. The training label is indicated by the box color (red = positive, blue = negative), and surrounding frames show temporal context. (c-d) were found automatically using our method to analyze the training data sets.

the set of classifiers that perform well on the current training data set, from active learning [25]. We show both theoretically and empirically that x' has a large influence on x if and only if most classifiers in the version space assign them the same labels. In general, estimating the version space is also prohibitively slow, but for the special case of boosting, we propose two faster algorithms.

Our definition of influence enables us to propose several new methods with which a user can diagnose the cause of a given misprediction. The main use we propose is to find and understand label noise in the training data set. Here, the user can determine whether errors (Figure 1(c)) or inconsistency (Figure 1(d)) in the training labels caused the misprediction by looking at the labels of the most influential training examples. Incorrectly or inconsistently labeled training data lead to significant prediction errors because the learning algorithm will find an overly complex rule that deviates from the true best classifier. Inconsistent labeling is common for tasks in which the class definitions are vague such as activity recognition (how far does a fly need to walk backwards to be considered "backing up" (Figure 1(d))), attribute learning (e.g. round vs. not round [11]), or labeling scenes (e.g. beach vs. coast [31]). Labeling mismatches are also found when labeling is done by different labelers, for example in crowdsourcing [30]. In these cases, showing similar previously labeled examples from the training set in real-time as the user is labeling an instance would ensure that labeling is consistent.

Second, the user can examine the magnitudes of the influence of all the training data on a mispredicted example. When no training examples had large influence on a mispredicted example, we observed that the training data lacked examples similar to the mispredicted test example. In this situation, the user can review the test example and add it and other similar examples to enrich the training data set. This will improve the training set's variability, and ensure all cases are adequately represented.

Third, focusing the engineer's attention on just the relevant training examples can give them insight into missing features in the current representation. If we find that the mispredicted test example is influenced by many examples of both classes in the training set, and that these examples are labeled correctly, then the current feature set may be insufficient for discriminating the two classes in this part of the data space. The engineer can examine these specific examples to devise useful features.

Our contributions are (1) proposing the use of influence between pairs of examples to analyze and understand a classifier's errors, (2) a practical distance metric that approximates influence for boosting and can be computed fast enough to be used interactively (Section 3), (3) several novel paradigms for using this metric to diagnose the cause of classification errors (Section 5), and (4) quantitative criteria to compare different distance metrics in estimating the influence (Section 5). In Section 5, we provide detailed validation of our method on a real-world machine learning system used by biologists. In Section 6, we validate our method on the ImageNet [8] object recognition data set.

2. Related work

There have been few formal efforts aimed at understanding a classifier's errors. The work most similar to ours is the idea of proximity proposed informally for random forests by [2]. For random forests, the proximity between two examples is defined as the number of trees for which two examples fall in to the same leaf node. This proximity measure was then used to find gross patterns via clustering and multi-dimensional scaling and to identify outliers in relatively small training data sets. However, it was not shown that close-by examples influence each other. In our work, we show that the training examples that most influence an example's predictions are close according to our distance metric.

[17] catalogued the failures of current object detection methods. The goal of their study was to help computer vision researchers understand the main limitations in object detection. The failures were analyzed by categorizing them, and studying which failure categories were dominant for different tasks. Their study was tied tightly to object detection methods and to the Pascal VOC dataset, while the goal in our work is to present a general method that can be applied by researchers to novel data sets and algorithms.

A few recent approaches [29, 32] have been developed for visualizing feature representations. Our work is complementary to these, as it is aimed at finding the training examples most responsible for specific classifier errors. It can be used to narrow down the set of examples to examine using these feature visualization approaches.

Several approaches attempt to improve robustness to label noise by attempting to automatically identify and down weight mislabeled examples, without any interaction with the user. In [24], the training data set is cleaned by automatically removing any training example with many examples with the opposite label nearby. In [13], the learning algorithm ignores examples that are difficult to predict. In [3], examples that are predicted differently by different algorithms are removed. All these methods require that the fraction of examples incorrectly labeled is very small and that there is sufficient training data to distinguish rare, highly informative, correctly labeled examples from mistakes. For example, training examples identified as mistakes by [3] would be identified as important examples to label by active learning approaches such as [26]. Our method can be used in addition to these methods which learn an accurate classifier to find, visualize and understand the context of, and remove the *source* of errors.

As discussed in the next section, our distance metric is heavily influenced by concepts developed for active learning. Central to our distance metric and active learning is the small set of classifiers that perform well on the current training data. For active learning, the example on which this set of classifiers disagree the most is considered most informative [26, 5, 21, 7, 1, 25]. These methods estimate how informative a *single* unlabeled example is based on its relationship to the current small set of relevant classifiers. We use this concept of the current set of reasonable classifiers to explore the relationship between *pairs* of examples.

3. Influence and classification-based dissimilarity

As discussed above, we define the influence of one example x' on another example x as the difference in prediction on x for the two classifiers trained with x' added with labels 1 and 0 to the training set $\mathcal{D}$. Formally, we define the influence of x' on x as:

$$J(x' \rightarrow x) \triangleq \mathbb{E}_h[h(x)|\mathcal{D}_1] - \mathbb{E}_h[h(x)|\mathcal{D}_0], \tag{1}$$

where $\mathcal{D}_1 = \mathcal{D} \cup \{(x', 1)\}$ and $\mathcal{D}_0 = \mathcal{D} \cup \{(x', 0)\}$ are the augmented training data sets, and the expected value is taken over the set of reasonable classifiers h that are learnable from the hypothesis space $\mathcal{H}$ given the training sets.

Our proposed dissimilarity measure to estimate this influence is inspired by the notion of version space. Version space is defined for perfectly separable data, and is the set of all classifiers $h \in \mathcal{H}$ that correctly partition the training data [5, 7]. More generally, we can instead consider the set of all reasonable classifiers that can be learned from small perturbations of the training data [1]. However, for this discussion we focus on the perfectly separable case.

We define the distance between two examples x and x' as the fraction of version space for which the predictions on x and x' disagree:

$$D(x, x') \triangleq \frac{1}{|\mathcal{V}|} \sum_{h \in \mathcal{V}} I(h(x) \neq h(x')), \tag{2}$$

where $\mathcal{V}$ is the version space for training set $\mathcal{D}$ and I is the indicator function (Figure 3(a-b)). Let us consider the simplified case in which the distribution of classifiers h given a training set, considered in Equation (1), is uniform among all classifiers in the version space of that training set. Then we can rewrite influence as

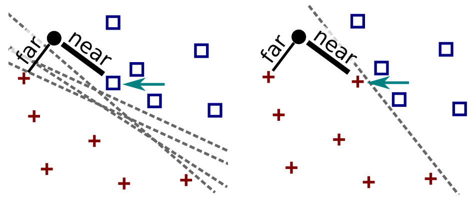
$$J(x' \rightarrow x) = \frac{1}{|\mathcal{V}_1|} \sum_{h \in \mathcal{V}_1} h(x) - \frac{1}{|\mathcal{V}_0|} \sum_{h \in \mathcal{V}_0} h(x), \tag{3}$$

where $\mathcal{V}_1$ and $\mathcal{V}_0$ are the restricted version spaces for training data sets $\mathcal{D}_1$ and $\mathcal{D}_0$, respectively.

FastBoot Algorithm

```
Given: $(x_1, y_1), \ldots, (x_k, y_k)$ where $x_u \in X, y_u \in Y = \{-1, +1\}$
Initialize weight distribution $D_1(u) = 1/m$.
For $j = 1, \ldots, M$:
    • Find $N$ best weak rules $h_j^1, h_j^2, \ldots, h_j^N$ with least errors $\epsilon_j^1, \epsilon_j^2, \ldots, \epsilon_t^N$ where $\epsilon_j^i = \Pr_{u \sim D_j}[h_j^i(x_u) \neq y_u]$
    • Assign weight $\alpha_j^i = F(\epsilon_j^i)/N$ to each $h_j^i$.
      (e.g. $F(\epsilon) = \frac{1}{2}\ln\left(\frac{1-\epsilon}{\epsilon}\right)$ for AdaBoost and $F(\epsilon) = (1 - 2\epsilon)$ for LogitBoost).
    • Update weight distribution $D_{j+1}(u) = \sum_{t=1}^{j} \sum_{i=1}^{k} G\big(\alpha_j^i y_u h_j^i(x_u)\big)/Z_j$, where $Z_j$ is a normalization factor. [e.g. $G(s) = (1 + e^s)^{-1}$ for LogitBoost]
Output the pool of weak rules:
    $S = \left\{(\alpha_j^i, h_j^i(x)) : 1 \leq j \leq M, 1 \leq i \leq N\right\}$
    $D(x, x') = \sum_{i=1}^{N} \sum_{j=1}^{M} \left|\alpha_j^i [h_j^i(x) \neq h_j^i(x')]\right|,$
```

Figure 2. FastBoot algorithm: A variation on the the boosting algorithm to learn a larger pool of weak rules.



(a) (b)

Figure 3. (a) Cartoon showing the intuition behind our version-space based distance metric. Red pluses indicate one class of training example, blue squares the other. Classifiers within the version space are shown by gray dashed lines. The black circle represents an unlabeled example. If we change the label of the blue square example indicated by the arrow to a red plus (b), the prediction of all classifiers in the version space will change for the unlabeled point. This is not the case for the red plus example indicated to be far.

We can rewrite our distance measure in Equation 2 as

$$D(x, x') = \frac{1}{|\mathcal{V}|} \sum_{h \in \mathcal{V}} \Big((1 - h(x))h(x') + h(x)(1 - h(x')) \Big)$$
$$= \frac{1}{|\mathcal{V}|} \sum_{h \in \mathcal{V}_1} (1 - h(x)) + \frac{1}{|\mathcal{V}|} \sum_{h \in \mathcal{V}_0} h(x)$$

If we assume that both classes are approximately equally likely for training example x' within $\mathcal{V}$: $|\mathcal{V}_1| \approx \frac{1}{2}|\mathcal{V}|$, then:

$$D(x, x') \approx \frac{1}{2} - \frac{1}{2}\left(\frac{1}{|\mathcal{V}_1|} \sum_{h \in \mathcal{V}_1} h(x) - \frac{1}{|\mathcal{V}_0|} \sum_{h \in \mathcal{V}_0} h(x) \right)$$
$$= \frac{1}{2}(1 - J(x' \to x))$$

The assumption $|\mathcal{V}_1| \approx \frac{1}{2}|\mathcal{V}|$ on x' implies that the training example x' is likely to be near the decision boundary. These are the examples in the training set that are most likely to contain errors, and what our method is focused on.

For practical cases, in which the training data is rarely separable and free of label noise, we approximate the ideal of version space by the set of all classifiers that have nearly the same accuracy as the current best classifier. We can sample classifiers from this set by bootstrapping, in which the training data is repeatedly subsampled in order to learn a large set of classifiers [21].

Our estimate of the distance between pairs of examples x and x' is thus

$$D(x, x') = \frac{1}{M} \sum_{i=1}^{M} \big| f_i(x) - f_i(x') \big| \quad (4)$$

where $f_1, f_2, \ldots, f_M$ are the classifiers learned using bootstrapping and $f_i(x)$ is the continuous prediction score for example x. We refer to this as the Bootstrap distance. This distance is general and is applicable to all classifiers. However, to estimate the distance using this is still slow because to get reasonable estimates of distance we will have to learn a large number of bootstrapped classifiers.

For boosting, the effective number of classifiers sampled can be increased by leveraging the internal mechanism of the classifier. The score given to an example is the weighted sum of weak rule predictions. Consider two pairs of examples that receive the same score: for the first pair most of the weak rules' predictions disagree, while for the second pair all weak rules agree. If we were to learn another classifier by subsampling the training data, it is more likely that this classifier will produce similar scores in the latter case. Using this intuition, we estimate the dissimilarity between two examples x and x' for boosting as:

$$D(x, x') = \frac{1}{M} \sum_{i=1}^{M} \sum_{j=1}^{T} \left| \alpha_j^i I(h_j^i(x) \neq h_j^i(x')) \right|, \quad (5)$$

where α_j^i is the weight of the j^{th} weak rule h_j^i of the i^{th} boosting classifier. We refer to this dissimilarity measure as BoostBoot dissimilarity.

For the case when decision stumps are used as weak rules in boosting, we developed an even faster method of estimating the dissimilarity (Figure 2). If we examine the BoostBoot dissimilarity (Equation 5), we see that it is a weighted sum over a large pool of weak rules. We modified the standard boosting algorithm to learn a larger pool of weak rules that approximate the set of weak rules selected by bootstrapping at each iteration. In each iteration of standard boosting, a weight distribution over the examples is generated. Then, the weak rule that has the least error over this weight distribution is picked and given a weight proportional to its accuracy. For our modified algorithm, we select the M best weak rules at each iteration. Assume that at any iteration, the distribution of example weights is similar across all the subsamples. Then, there will be a large overlap between the set of M best weak rules selected at iteration j using all the training data and the M weak rules selected individually at iteration j across the M bootstrap subsamples. We refer to this as the FastBoot dissimilarity. Note that learning this single classifier in which we select MT weak rules takes the same amount of time as learning a single classifier with standard boosting.

4. Implementation

The FastBoot dissimilarity is fast and practical to compute. In our implementation, we modified the LogitBoost [15] algorithm to learn weak rules for the FastBoot dissimilarity. In our MATLAB implementation, learning takes 22 seconds for a training set with 11,407 examples and 5439 features (dual quad-core machine with 24 GB of RAM). Computing dissimilarity from a particular example to 1,000 examples takes just 2 seconds, thus our dissimilarity measure is fast enough for interactive use.

We have integrated FastBoot into an open-source, interactive machine learning system used by biologists to train animal behavior classifiers, called JAABA [19]. Our dissimilarity measure is particularly well-suited to JAABA because (1) JAABA is an interactive system in which users iteratively label and retrain classifiers, thus information provided by our dissimilarity measure can easily be used, (2) JAABA is used by biologists unfamiliar with machine learning who thus find it difficult to understand the cause of classifier failures, and (3) consistently labeling behavior is difficult because the boundaries of behavior categories are fuzzy (Section 5).

5. Application to fly behavior classification

We applied our dissimilarity measure to a large, diverse fly-behavior data set consisting of $>$20,000 15-minute videos of groups of 20 fruit flies [19]. It contains data from $>$2,000 behaviorally diverse genetic lines of flies. We used 12 labeled data sets corresponding to 12 locomotion and social behaviors. For each behavior class, the training data consisted of 1,300-26,000 labeled frames sampled from several animals, videos, and genotypes. The feature representations varied between 5,000 and 21,000-dimensional. Note that JAABA allows the user to interactively add labels and retrain the classifier, thus the data are not i.i.d. The data was most often labeled in contiguous intervals termed bouts. As examples within a bout are similar, whenever we split the training data into hold-out and training sets, we split the data at the bout-level.

Behavior	Bootstrap	BoostBoot	FastBoot	Boosting	L1	LDA
Att. Cop.	0.867	**0.888**	0.871	**0.877**	0.559	0.477
Backup	0.846	**0.890**	**0.889**	0.886	0.759	0.491
Chase	0.820	**0.907**	**0.898**	0.895	0.678	0.476
Crabwalk	0.877	0.920	**0.921**	0.913	0.686	0.500
Jump	0.804	**0.853**	**0.851**	0.847	0.807	0.493
Stop	0.773	**0.845**	0.841	**0.843**	0.622	0.480
Walk	0.748	0.926	0.944	**0.947**	0.815	0.498
Ctr.-pivot	0.937	0.945	0.947	**0.950**	0.896	0.446
Tail-pivot	0.876	**0.898**	**0.893**	**0.893**	0.762	0.495
Touch	0.847	**0.886**	**0.885**	0.875	0.612	0.509
Wing Ext.	0.912	**0.949**	**0.948**	0.940	0.769	0.507
Wing Flick	0.848	**0.895**	**0.890**	0.885	0.707	0.507
Average	0.846	**0.900**	**0.898**	0.896	0.723	0.490
Complexity	500	50	1	1	0	1

Table 1. Comparison of area under the ROC curves for different methods for identifying influential training examples on 12 different behavior classification tasks. Distance metrics proposed in this work are shown in white columns, while existing distance metrics are shown in gray. Complexity refers to the number of classifiers that were trained to obtain each number. We highlight the best performing algorithm and the best performing efficient algorithm in bold.

5.1. Finding influential examples

In this section, we compare how well different distance metrics can find the training examples that have influence on a given test example. For each sampled pair consisting of one labeled training example and one unlabeled test example, we estimated whether the training example has large influence on the test example, in the sense of Eq. 1. We then used this as the ground-truth to which a suite of distance metrics are compared.

To measure whether training examples had influence beyond noise on a test example in a computationally feasible manner, we made a few modifications to the definition in Eq. 1 (see Supplementary Materials). For each of 12 different animal behavior data sets, we sampled 760 training examples and computed their influence on an unlabeled test set of 1140 examples (thus, we consider 866,400 pairs).

Once we had created our ground-truth labels of influence for pairs of examples, we then determined how well we can replicate this classification by thresholding various distance metrics. We used the Receiver Operator Characteristic (ROC) Curve to plot the performance of the different distance metrics in predicting the influence of a training example on a test example (Table 1). We compared the following distances. Bootstrap (Equation (4)), BootBoost (Equation (5)) and FastBoot (Figure 2) are the distance metrics we define in Section 3. We also compared to the Boosting distance metric, in which the distance is computed similar to the BootBoost/FastBoot distance metrics, but using just a single classifier learned from all the training data. For

Behavior	FastBoost	L1	Random	N. examples
Att. Copulation	**0.33**	0.26	0.14	700
Backup	**0.58**	0.51	0.14	460
Chase	**0.23**	0.17	0.06	477
Crabwalk	**0.77**	0.58	0.23	176
Jump	**0.43**	0.40	0.12	300
Stop	**0.45**	0.33	0.10	293
Walk	**0.41**	0.33	0.08	407
Center-pivot	**0.71**	0.56	0.19	174
Tail-pivot	**0.30**	0.27	0.08	671
Touch	**0.62**	0.55	0.17	137
Wing Extension	**0.68**	0.56	0.17	64
Wing Flick	**0.67**	0.52	0.22	180

Table 2. For each of 12 behavior data sets, we report the improvement in classifier prediction score on mispredicted hold-out examples after flipping the label of the 25 closest training examples, as determined by FastBoot, or, for comparison, L1-distance or randomly. N. examples indicates the number of mispredicted test examples we analyzed.

comparison, we also computed the normalized L1 distance (each feature was normalized by its standard deviation in the training set). Finally, we compared to the distance in the 1-dimensional linear subspace defined by linear discriminant analysis (LDA).

LDA-based distance performed the worst in all cases. We believe this is because distances in the 1-D projection defined by LDA are akin to classification scores and thus it selects examples with the most similar score. Normalized L1 distance performed second worst in all cases, most likely because it is affected by irrelevant features. Bootstrap distance also performed relatively poorly, most likely because the number of classifiers trained using bootstrapping (we sampled 500) was insufficient. FastBoot, BootBoost, and (somewhat surprisingly) Boosting all performed well. FastBoot is 50X faster than BootBoost, and has a slight performance edge over Boosting. As FastBoot is the best efficient approximation of influence, in the next experiments, we show that FastBoot can be helpful in understanding a classifiers errors.

5.2. Effect of influential examples

In this experiment, we show that, not only can we find influential examples, but changing the labels of the selected training examples can improve the classifier's performance. For this experiment, for each of the 12 animal behavior data sets, we split the training data into training (95%) and hold-out (5%) sets and we found examples that were mispredicted in the hold-out set. For each of these, we found the 25 closest training examples with opposite label using the FastBoot distance metric. We then flipped the labels of these training examples, retrained a new classifier and measured the improvement in scores of the mispredicted example. We normalized the improvement in scores by the 80th percentile score on the training set, thus -1 and 1 correspond to high confidence predictions. For comparison, we measured the improvement using two baseline methods, one in which the nearest training examples are chosen based on L1-distance and the other in which they are chosen randomly. For all behaviors, the median improvement using our method is much larger than with the baseline methods (Table 2).

5.3. Finding inconsistently labeled training data in practice

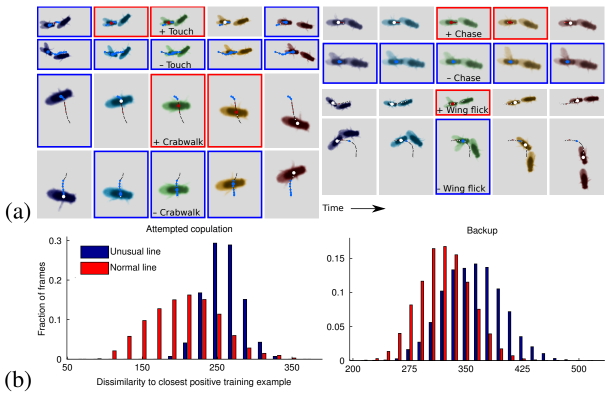


Figure 4. (a) Similar looking examples with opposite labels, plotted as Figure 1(c-d). The center frame shows the two training examples with opposite labels that are closest according to the FastBoot distance (two more are shown in Figure 1(c-d)). (b) Distribution of dissimilarity of predicted positives to the training data for two behavior data sets. We show the distribution for an "unusual" fly line for which the classifier performs poorly and a "normal" fly line for which the classifier performs well.

In this section, we show that our FastBoot algorithm can be combined with an interactive learning framework to allow a user to curate inconsistently labeled data.

As illustrated in Figure 1 as well as by the inter-annotator disagreement rates reported in e.g. [4], the boundaries between behavior categories can be fuzzy. It is often difficult for even a single annotator to consistently label behaviors. We found that this was the case when categorizing the subtly different ways walking flies can turn. As we labeled "body-turns", in which the fly pivots after stopping, we realized that our definition of the behavior drifted over time, and that we had labeled very similar bouts with opposite labels.

We created a module for JAABA that allowed the user to choose an unlabeled frame (e.g. one that was being mispredicted by the classifier) and query for the closest training examples with both positive and negative labels. The user could then use JAABA to correct any mislabeled training examples. Alternatively, they could also query for closest unlabeled examples, which they could then label and add to the training set. We used this module and JAABA to curate our initial "body-turn" training data set. In most cases, we found a mislabeled training example within the 20 closest frames to the mispredicted test example. During curation, 95 labeled examples were removed and 125 were added from an initial training set containing 2854 examples.

To compare the accuracy of the curated and initial classifiers, we created a ground-truth data set by manually labeling 6,000 frames in novel test movies for which the two classifiers disagreed (note that these frames are often the difficult cases, thus accuracy rates here do not reflect the global accuracy). For frames labeled (211 positive, 2109 negative frames), the error rate improved from **80% to 20%** after curation (false negative rate improved from 83% to 17%, false positive rate went from 47% to 53%).

We found that a similar type of curation may be helpful for the other 12 behavior data sets. For each data set, we found the pair of training examples with opposite labels that had the smallest dissimilarity. These closest examples are shown in Figure 4(a) and are visually very similar.

5.4. Detecting feature set limitations

Among all the fly behavior data sets, the "righting detector" was the most complex, consisting of over 26,000 training examples of dimensionality 21,000. For this data set, the training error was non-zero (171 training examples labeled as negative were classified as positive). We used our dissimilarity measure to determine that, in this case, our feature representation was lacking. We clustered the false positive training examples using our dissimilarity measure and average-linkage agglomerative clustering. We then found the 10 true positive training examples nearest to the largest cluster, which contained 27 examples. We found that there was not a single feature of the set of 21,000 for which a single threshold could correctly separate these data (at least 8 mistakes were made by each weak learner). Thus, using our dissimilarity measure, we were able to find, visualize, and understand a particularly difficult part of the large data space, finding concrete examples for which the current representation is feature-limited.

5.5. Searching for unusual test data

The assumption in most machine learning algorithms is that the training and test data come from the same distribution. This is often not the case when a classifier is used in practice [28, 22]. For example, the fly behavior data set consists of 2,200 behaviorally diverse genotypes, and it would be prohibitive to train on all of them. In the biology application of comparing the behavior of different genotypes, it is important that the accuracy of the classifier be similar on all genotypes. We show that our dissimilarity measure can be used to find "unusual" unlabeled data subsets for which the classifier performs poorly. Data from these sets can then be added to the training data set to improve the classifier.

For each of two behavior classifiers ("attempted copulation" and "backup"), we examined all fly lines for which the classifier predicted an extremely high number of positives. We manually selected two lines from this set, one

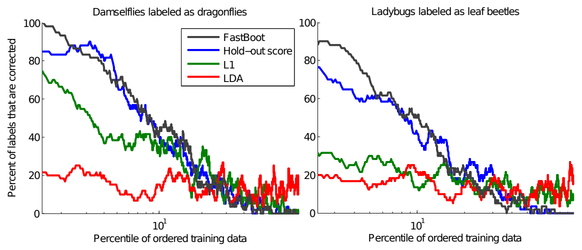


Figure 5. Percent of leaf beetle/dragonfly images that are truly ladybugs/damselflies, when sorted by 1) median dissimilarity to the ladybug/damselfly data set (sliding window of size 61), 2) cross-validation scores, 3) median L1 distance to the ladybug/damselfly data set and 4) median LDA distance to ladybug/damselfly data set.

which truly performed the behavior more ("normal") and the other that appeared to have a high number of mispredictions due to that line's "unusual" behavior (e.g. flies that jumped a lot). We examined the distribution of the dissimilarity from all positive predictions to the closest positive training example. As shown in Figure 4(b), a larger fraction of examples in the "unusual" line were far from all the training data. Thus, unusual data sets for which a classifier's generalization performance is poor can be found by examining those that have a higher proportion of examples far from the training set.

6. Application to the ImageNet data set

6.1. Removing label noise in crowdsourced annotations

The large ImageNet data set, consisting of 500-1,000 images for each of >20,000 object categories, was annotated using crowdsourcing [8]. While these manual annotations are accurate at higher levels in the category hierarchy, label errors are more pronounced for expert-level tasks, such as distinguishing families or species of insects. In this section, we show how our dissimilarity measure can be used to find mislabeled training data collected using crowdsourcing techniques, for example for correction by an expert annotator.

We restricted our analysis to the 28 categories below the "insect" category in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2010 data set. This is because, to find truly mislabeled examples, we required a human with expert knowledge to manually relabel the ImageNet data. We used the DeCAF deep network trained on the 1000-category ILSVRC 2012 data set [9] to compute a 4096-dimensional representation of each image. We used the same GentleBoost algorithm, as above, to learn 28 one-versus-all classifiers.

By looking at mispredicted examples in the validation data set and their nearest neighbors (according to our dis-

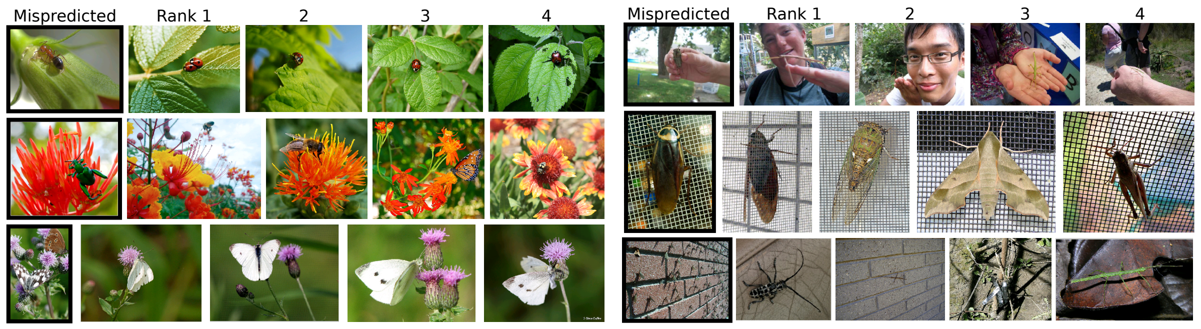


Figure 6. Mispredicted validation images (black boxes) in context of their 4 nearest neighbors, according to our dissimilarity measure.

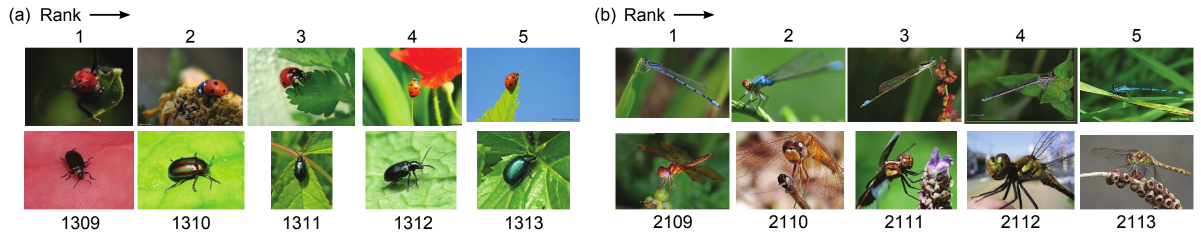


Figure 7. (a) Leaf beetle training images sorted by their dissimilarity to the ladybug data set. Top row shows the 5 closest examples, bottom row the 5 farthest examples. (b) Dragonfly training images sorted by their dissimilarity to the damselfly data set.

similarity measure) in the training set (Figure 6), we identified two types of common, systematic errors: ladybugs were often labeled as leaf beetles, and damselflies were often labeled as dragonflies. We used our FastBoot algorithm to compute the median dissimilarity from each leaf beetle training image to all ladybug training images, and, similarily, the median dissimilarity from each dragonfly training image to all damselfly training images. We then sorted the leaf beetle/dragonfly training data sets according to this median dissimilarity, and a volunteer with entomology expertise went through these training images in this order, and labeled true ladybugs/damselflies. For comparison, we also sorted the dataset according to median normalized L1 distance, median LDA distance. Finally, we also used cross-validation scores for sorting. Our dissimilarity indeed puts true ladybugs/damselflies at the beginning of the list (Figure 7) and performs much better than the L1 and LDA distance metrics (Figure 5). Sorting by cross-validation performs only slightly worse on this particular task. As described in the next section, the main advantage of our method over cross-validation is that it can provide context to a given error. That is, when we ask a user to review an example, we can also show them the closest examples.

6.2. Gaining insight into complex classifiers

Examining mispredicted examples in our validation set in the context of their nearest neighbors can give us insight into what properties of the data the classifier has learned. Figure 6 shows selected mispredicted examples and their nearest neighbors within the 28-category training data set. In all of these cases, the commonality between the mispredicted example and its neighbors is the image background. Anecdotally, this suggests that the classifier is making strong use of the image background, which captures information about what type of environment the insect was found in. In many cases, this is a sensible feature to use, as many insects are found only in specific habitats. An interesting question for further research is whether it is helpful to improve invariance to the image background, e.g. with the use of saliency maps [18]. Combining this technique for selecting important images with the approach of [32] for visualizing convolutional network feature representations of a given example may also be fruitful.

7. Discussion

Increases in both data-set size and complexity of classifier space searched have made it difficult for engineers to look at and understand both the data and the resulting classifiers. In this work, we show that our influence-based distance metric can be used in multiple ways to analyze, understand, and improve the current configuration of classifier family, features and training data (note that our definition of influence is a function of this joint configuration, and does not necessarily apply to any single component alone).

With this method, engineers have access to the intuitive methods for understanding nearest-neighbor classifiers without sacrificing the advantages in complexity and generalization of modern classifiers. This method will be of use to machine learning experts, engineers, and data scientists developing computer vision and machine learning systems for real-world applications.

8. Acknowledgements

We would like to thank Aljoscha Nern for providing expertise on insect identification, and Yoav Freund, Nakul Verma, Steve Branson, Daniel Hsu, Charles Elkan, Fernando Amat, and Philipp Keller for valuable discussions.

References

[1] M.-F. Balcan, A. Beygelzimer, and J. Langford. Agnostic active learning. *Journal of Computer and System Sciences*, 75(1):78–89, 2009.

[2] L. Breiman. Looking inside the black box. *Wald Lecture II, UC Berkeley*, 2002.

[3] C. E. Brodley and M. A. Friedl. Identifying mislabeled training data. *arXiv preprint arXiv:1106.0219*, 2011.

[4] X. P. Burgos-Artizzu, P. Dollár, D. Lin, D. J. Anderson, and P. Perona. Social behavior recognition in continuous video. In *CVPR*, 2012.

[5] D. Cohn, L. Atlas, and R. Ladner. Improving generalization with active learning. *Machine Learning*, 15(2):201–221, 1994.

[6] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[7] S. Dasgupta. Coarse sample complexity bounds for active learning. *NIPS*, 2006.

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. *CVPR*, 2009.

[9] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. *arXiv preprint arXiv:1310.1531*, 2013.

[10] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results, 2012.

[11] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth. Describing objects by their attributes. In *CVPR*, 2009.

[12] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: an incremental bayesian approach tested on 101 object categories. In *CVPR*, 2004.

[13] Y. Freund. A more robust boosting algorithm. *arXiv preprint arXiv:0905.2138*, 2009.

[14] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55:119–139, 1997.

[15] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *The Annals of Statistics*, 28(2):337–407, 2000.

[16] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. Technical report, California Institute of Technology, 2007.

[17] D. Hoiem, Y. Chodpathumwan, and Q. Dai. Diagnosing error in object detectors. In *ECCV*, 2012.

[18] X. Hou and L. Zhang. Saliency detection: A spectral residual approach. In *CVPR*, pages 1–8, 2007.

[19] M. Kabra, A. A. Robie, M. Rivera-Alba, S. Branson, and K. Branson. JAABA: Interactive Machine Learning for Automatic Annotation of Animal Behavior Detection. *Nature Methods*, 10(1), 2013.

[20] T. Kraska, A. Talwalkar, J. Duchi, R. Griffith, M. J. Franklin, and M. Jordan. MLbase: A distributed machine-learning system. In *Conference on Innovative Data Systems Research*, 2013.

[21] N. A. H. Mamitsuka. Query Learning Strategies using Boosting and Bagging. In *ICML*, 1998.

[22] S. J. Pan and Q. Yang. A survey on transfer learning. *TKDE*, 22(10):1345–1359, 2010.

[23] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. Labelme: a database and web-based tool for image annotation. *IJCV*, 2008.

[24] J. Sánchez, R. Barandela, A. Marqués, R. Alejo, and J. Badenas. Analysis of new techniques to obtain quality training sets. *Pattern Recognition Letters*, 24(7):1015–1022, 2003.

[25] B. Settles. Active learning literature survey. *University of Wisconsin, Madison*, 2010.

[26] H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Comp. Learning Theory*, 1992.

[27] C. Sommer, C. Straehle, U. Kothe, and F. A. Hamprecht. iLastik: Interactive learning and segmentation toolkit. In *Biomedical Imaging*, pages 230–233, 2011.

[28] A. Torralba and A. A. Efros. Unbiased look at dataset bias. In *CVPR*, pages 1521–1528, 2011.

[29] C. Vondrick, A. Khosla, T. Malisiewicz, and A. Torralba. Hoggles: Visualizing object detection features. In *ICCV*, pages 1–8, 2013.

[30] P. Welinder, S. Branson, S. Belongie, and P. Perona. The multidimensional wisdom of crowds. *NIPS*, 2010.

[31] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. SUN database: Large-scale scene recognition from abbey to zoo. In *CVPR*, 2010.

[32] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, pages 818–833. 2014.