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.

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.

We propose a method to gain an understanding of a classifier's errors for a given data set hinged 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. The influence of a training example x' on a test example x can be found in a straightforward manner 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.

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.

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 or inconsistency in the training labels caused the misprediction by looking at the labels of the most influential training examples. 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.

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 dissimilarity metric that approximates influence.

Formally, we define the influence of x' on x as:

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

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.

To define the dissimilarity metric we borrow the idea of *version space*, the set of classifiers that perform well on the current training data set, from active learning [3]. 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 (Figure 1(a)):

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

where $\mathcal{V}$ is the version space for training set $\mathcal{D}$ and I is the indicator function.

This is an extended abstract. The full paper is available at the Computer Vision Foundation webpage.

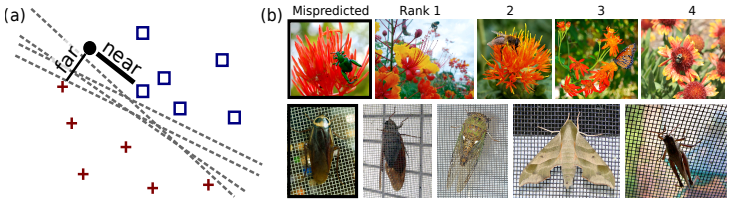

Figure 1: (a) Cartoon showing the intuition behind our version-space based distance metric. Red pluses indicate one class of training example, blue squares the other. Gray dashed lines indicate classifiers in the version space. The black circle represents an unlabeled example. If we change the label of the "near" example, the prediction of many classifiers in the version space will change for the unlabeled point. This is not the case for the "far" example. (b) Mispredicted validation images (black boxes) from the ImageNet dataset using DeCaf features [2] in context of their 4 nearest neighbors, as selected using FastBoot. These examples suggest that the convolutional network is making strong use of the image background.

We show that our version-space distance and influence are closely related to each other:

$$D(x,x') \approx \frac{1}{2}(1 - J(x' \rightarrow x))$$

To compute $D(x,x')$ in practice, we approximate 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. For boosting in particular, we developed an approximation to bootstrapping, FastBoot, which is fast enough for interactive use. That is, a user can select an example and query for the closest training examples in real time. For a training set consisting of 11,407 examples with 5,439 features, learning the parameters of our FastBoot dissimilarity required a one-time cost of 22 seconds, after which computing the dissimilarity from any selected example to 1,000 other examples takes just 2 seconds.

We empirically showed that our FastBoot dissimilarity measure is a good approximation of influence, and can find the most influential examples better than baseline approaches including the L1-distance as well as linear discriminant analysis. We also showed that changing the labels of these selected neighbors in the training data does have a large impact on the prediction of the original test example, again much larger than for training neighbors selected randomly or based on the L1 distance. We also showed several practical uses of our FastBoot dissimilarity. First, we integrated it into an interactive machine learning system for training behavior classifiers, and showed that it could help a user identify and remove label noise and greatly improve the generalization performance of the classifier. By using FastBoot on the ImageNet [1] data set, we showed how it could be used to identify label noise in crowd-sourced annotations of large data sets. We also showed that our dissimilarity measure could be used to identify both feature set limitations and regions of the data space where training data is needed. Finally, we showed how FastBoot can be used to gain insight into complex classifiers such as deep convolutional networks (Figure 1(b)).

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

[2] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. *arXiv preprint arXiv:1310.1531*, 2013.

[3] Burr Settles. Active learning literature survey. *University of Wisconsin, Madison*, 2010.