Debug ging Machine Learning T asks ar X iv : 16 03 . 07 292 v 1 [ cs . LG ] 23 Mar 2016 Ale ks and ar Chak arov A , Ad ity a Nor i B , Sr ir am Raj am ani B , Shay ak Sen C , and Deep ak Vij ay ke er thy D A University of Colorado , Boulder B Microsoft Research C Carnegie Mellon University D IBM Research March 24 , 2016 Abstract Unlike traditional programs ( such as operating systems or word processors ) which have large amounts of code , machine learning tasks use programs with relatively small amounts of code ( written in machine learning libraries ) , but vol umin ous amounts of data . Just like developers of traditional programs debug errors in their code , developers of machine learning tasks debug and fix errors in their data . However , algorithms and tools for debugging and fixing errors in data are less common , when compared to their counterparts for detecting and fixing errors in code . In this paper , we consider classification tasks where errors in training data lead to mis class ifications in test points , and propose an automated method to find the root causes of such mis class ifications . Our root cause analysis is based on Pearl s theory of causation , and uses Pearl s PS ( Pro b ability of Suff iciency ) as a scoring metric . Our implementation , P si , enc odes the computation of PS as a prob abil istic program , and uses recent work on prob abil istic programs and transformations on prob abil istic programs ( along with gray - box models of machine learning algorithms ) to efficiently compute PS . P si is able to identify root causes of data errors in interesting data sets . 1 Introduction Machine learning techniques are used to perform data - driven decision - making in a large number of diverse areas including image processing , medical diagnosis , credit decisions , insurance decisions , email spam detection , speech recognition , natural language processing , robotics , information retrieval and online advertising . Over time , these techniques have been hon ed and tuned , and are now at a stage where machine learning libraries [ 1 , 2 ] are used as black - boxes by 1 program mers with little or no expertise in the details of the machine learning algorithms themselves . The black - box nature of the reuse , however , has an unfortunate downside . Current implementations of machine learning techniques provide little insight into why a particular decision was made . Because of this absence of transparency , debugging the outputs of a machine learning algorithm has become incredibly hard . Most programmers who implement machine learning use libraries to build models from vol umin ous training data , and then use these models to perform predictions . These machine learning libraries often employ complex , sto ch astic , or approximate , search and optimization algorithms that search for an optimal model for a given training data set . The model is then applied to a set of unseen test samples in the hope of satisfactory general ization . When general ization fails , i . e ., an incorrect result is produced for a test input , it is often difficult to debug the cause of the failure . Such failures can arise due to several reasons . Common causes for failure include bugs in the implementation of the machine learning algorithm , incorrect choice of features , incorrect setting of parameters ( such as degree of the po lyn omial for regression or number of layers in a neural network ) when invoking the machine learning library , and noise in the training set . Over time , bugs in implementation of machine learning algorithms get detected and fixed . There is a lot of work in feature selection [ 3 ], and parameter choices can be made by systematically building models for various parameter values and choosing the model with the best validation score [ 4 ]. However , since training data is typically vol umin ous , errors in training data are common and notoriously difficult to debug . This suggests a new class of debugging problems where programs ( machine learning class ifiers ) are learnt from data and bugs in a program are now the result of faults in the data . In this paper , we focus on debugging machine learning tasks in the presence of errors in training data . Specifically we consider classification tasks , which are typically implemented using algorithms such as log istic regression [ 5 ] and boosted decision trees [ 6 ]. Suppose we train a class ifier on training data ( which has errors ), and the class ifier produces incorrect results for one or more test points . We desire to produce an automated procedure to identify the root cause of this failure . That is , we would like to identify a subset of training points that influences the classification for these test points the most . Therefore , correcting mistakes in these training points is most likely to fix the incorrect results . Our algorithm for identifying root causes is inspired by the structural equations framework of causation , as formulated by Jude a Pearl [ 7 , 8 ]. We think of each of the training data points as possible causes of the mis class ification in the test data set , and calculate for each such training point , a score corresponding to how likely it is that the current label for that point is the cause for the mis class ification of the test data set . A simple measure of the score of a training point can be obtained by merely flipping the label of the training point and observing if the flip improves the results of the class ifier on test points . However , such a simple measure does not work when errors exist in several training points , and several training points together cause the incorrect results in the test points . Thus , the score we calculate for each training point t considers 2 altern ate counter fact ual worlds , where training points are labeled with several possible values ( other than the value in the training data ), and sums up the probability that flipping the label of t causes the mis class ification error in the test data , among all such alternate worlds . In Pearl s framework , such a score is called the probability of su fficiency or PS for short . One of the main difficulties in calculating the probability of su fficiency is that the class ifier ( or model ) needs to be rele ar nt for alternate worlds . Each of these model computing steps ( also called as training steps ) is expensive . We use a gray box view of the machine learning library , and profile key intermediate values ( that are hand - picked for each machine learning algorithm ) during the initial training phase . Using these values , we build a gray - box abstraction of the training process by which the model for a new training set ( which is obtained by flipping certain number of training labels ) can be obtained efficiently without the need to perform complete ( and expensive ) ret raining . Finally , we are able to am ort ize the cost of computing the PS score by sharing common work across the computation for different training points . In order to carry out these optimizations , we model the PS computation as a prob abil istic program [ 9 ]. Prob abil istic programs allow us to represent all of the above optimizations such as using gray - box models , using instrument ed values from actual training runs , and sharing work across multiple PS comput ations as program transformations . We are also able to leverage recent progress in efficient inference of prob abil istic programs to scale the computation of PS scores to large data sets . We have implemented our root cause detection algorithm in a tool P si . P si currently works with two popular class ifiers : ( 1 ) log istic regression , and ( 2 ) boosted decision trees . For these class ifiers , P si runs a production quality implementation of the techniques , profiles specific values and builds an abstract gray - box model of the class ifier , which avoids expensive re - training . Armed with this gray - box model , P si performs scalable inference to compute the PS values for all points in the training set . P si is able to identify root causes of mis class ifications in several interesting data sets . In summary , the main contributions of this paper are as follows : We propose using the structural equations framework of caus ality , and specifically Pearl s PS score to compute root causes of failures in machine learning algorithms . We model the PS computation as a prob abil istic program , and this enables us to leverage efficient techniques developed to perform inference on prob abil istic programs to calculate PS scores . We build gray - box models of the machine learning techniques by profiling actual training runs of the library , and using prof iled values to build abstract models of the training process . We am ort ize work across PS comput ations of different training points . Prob abil istic programs allow us to carry out these optimizations and reason about them as program transformations . We have built a tool P si implementing the approach for log istic regression 3 Training Phase Training Set ML Training Al gorithm Test Set · Class ifier Class Lab els { 1 , 1 } Evaluation Phase Figure 1 : A two - stage design flow of a machine learning task : training phase in which the ML algorithm A is applied to training set to learn class ifier h , and evaluation phase to judge the quality of h on test set . and boosted decision trees . P si is able to identify root causes of mis class ifications in several interesting data sets . We hypothes ize that this approach can be generalized to other machine learning tasks as well . 2 Overview We motivate our approach through the experience of Alice , a typical developer who uses machine learning . 2 . 1 Typical Sc enario Alice is not a machine learning expert , but needs to write a class ifier for images of vehicles and animals . Mall ory is a machine learning expert who built a classification library using state of the art machine learning techniques . Alice decides to use Mall ory s library , and since machine learning libraries are driven by data , she carefully collects some amount of training data { xi } i with images of cats , dogs , elephants trucks , cars , buses etc ., with labels y i = 1 or y i = 1 , stating whether an image is that of a vehicle or an animal respectively . She partitions it into a training set = { ( xi , y i ) } M i = 1 , and a test set , and picks out her favorite ML algorithm , log istic regression , to learn a binary class ifier that separates vehicles from animals . Alice runs Mall ory