Noname manuscript No.
(will be inserted by the editor)

An Instance Level Analysis of Data Complexity

Michael R. Smith · Tony Martinez · Christophe Giraud-Carrier

Received: date / Accepted: date

Abstract Most data complexity studies have focused on characterizing the complexity of the entire data set and do not provide information about individual instances. Knowing which instances are misclassified and understanding why they are misclassified and how they contribute to data set complexity can improve the learning process and could guide the future development of learning algorithms and data analysis methods. The goal of this paper is to better understand the data used in machine learning problems by identifying and analyzing the instances that are frequently misclassified by learning algorithms that have shown utility to date and are commonly used in practice. We identify instances that are hard to classify correctly (*instance hardness*) by classifying over 190,000 instances from 64 data sets with 9 learning algorithms. We then use a set of hardness measures to understand why some instances are harder to classify correctly than others. We find that class overlap is a principal contributor to instance hardness. We seek to integrate this information into the training process to alleviate the effects of class overlap and present ways that instance hardness can be used to improve learning.

Keywords Instance hardness · Dataset hardness · Data complexity

1 Introduction

It is widely acknowledged in machine learning that the performance of a learning algorithm is dependent on both its parameters and the training data. Yet, the bulk of algorithmic development has focused on adjusting model parameters without fully understanding the data that the learning algorithm is modeling. As such, algorithmic development for classification problems has largely been measured by classification accuracy, precision, or a similar metric on benchmark data sets. These metrics, however, only provide aggregate information about the learning algorithm and the task upon which it operates. They fail to offer any information about which instances are misclassified, let alone why they are misclassified. There is some speculation as to why some instances are misclassified, but, to our knowledge, no thorough investigation (such as the one presented here) has taken place.

Michael R. Smith · Tony Martinez · Christophe Giraud-Carrier
Department of Computer Science, Brigham Young University, Provo, UT, 84602
E-mail: msmith@axon.cs.byu.edu

Previous work on instance misclassification has focused mainly on isolated causes. For example, it has been observed that outliers are often misclassified and can affect the classification of other instances (Abe et al, 2006). Border points and instances that belong to a minority class have also been found to be more difficult to classify correctly (Brighton and Mellish, 2002; van Hulse et al, 2007). As these studies have had a narrow focus on trying to identify and handle outliers, border points, or minority classes, they have not generally produced an agreed-upon definition of what characterizes these instances. At the data set level, previous work has presented measures to characterize the overall complexity of a data set (Ho and Basu, 2002). Data set measures have been used in meta learning (Brazdil et al, 2009) as well as to understand under what circumstances a particular learning algorithm will perform well (Mansilla and Ho, 2004). As with the performance metrics, the data complexity measures characterize the overall complexity of a data set but do not look at the instance level and thus cannot say anything about why certain instances are misclassified. It is our contention that identifying which instances are misclassified and understanding why they are misclassified can lead to improvements in machine learning algorithm design and application.

The misclassification of an instance depends on the learning algorithm used to model the task it belongs to and its relationship to other instances in the training set. Hence, any notion of *instance hardness*, i.e., the likelihood of an instance being misclassified, must be a relative one. However, generalization beyond a single learning algorithm can be achieved by aggregating the results from multiple learning algorithms. We use this fact to propose an empirical definition of instance hardness based on the classification behavior of a set of learning algorithms that have been selected because of 1) their diversity, 2) their utility, and 3) their wide practical applicability. We then present a thorough analysis of instance hardness, and provide insight as to why hard instances are frequently misclassified. To the best of our knowledge our research is the first at reporting on a systematic and extensive investigation of the issue.

We analyze instance hardness in over 190,000 instances from 64 classification tasks classified by nine learning algorithms. We find that a considerable amount of instances are hard to classify correctly–17.5% of the investigated instances are misclassified by at least half of the considered learning algorithms and 2.3% are misclassified by all of the considered learning algorithms. Seeking to improve our understanding of why these instances are misclassified becomes a justifiable quest. To discover why these instance are hard to classify, we introduce a set of measurements (*hardness measures*). The results suggest that class overlap has the strongest influence on instance hardness and that there may be other features that affect the hardness of an instance. Although we focus on hardness at the instance level, the measures can also be used at the data set level by averaging the values of the instances in the data set. Further, we incorporate instance hardness into the learning process by modifying the error function of a multilayer perceptron and by filtering instances. These methods place more emphasis on the non-overlapping instances, alleviating the effects of class overlap. We demonstrate that incorporating instance hardness into the learning process can significantly increase classification accuracy.

The remainder of the paper is organized as follows. In Section 2, we introduce and define instance hardness as an effective means of identifying instances that are frequently misclassified. The hardness measures are presented in Section 3 as a means of providing insight into why an instance is hard to classify correctly. Section 4 presents the experimental methodology. An analysis of hardness at the instance level is provided in Section 5 followed by Section 6 which demonstrates that improved accuracy can follow from integrating instance hardness into the learning process. Section 7 compares instance hardness at the data set level with previous data set complexity studies. Section 8 provides related works and Section 9 concludes the paper.

2 Instance Hardness

Our work posits that each instance in a data set has a hardness property that indicates the likelihood that it will be misclassified. For example, outliers and mislabeled instances are expected to have high instance hardness since a learning algorithm will have to overfit to classify them correctly. Instance hardness seeks to answer the important question of what is the probability that an instance in a particular data set will be misclassified.

As most machine learning research is focused on the data set level, one is concerned with maximizing $p(h|t)$, where $h : X \to Y$ is a hypothesis or function mapping input feature vectors X to their corresponding label vectors Y, and $t = \{(x_i, y_i) : x_i \in X \wedge y_i \in Y\}$ is a training set. With the assumption that the pairs in t are drawn i.i.d., the notion of instance hardness is found through a decomposition of $p(h|t)$ using Bayes' theorem:

$$\begin{aligned} p(h|t) &= \frac{p(t|h)\; p(h)}{p(t)} \\ &= \frac{\prod_{i=1}^{|t|} p(x_i, y_i|h)\; p(h)}{p(t)} \\ &= \frac{\prod_{i=1}^{|t|} p(y_i|x_i, h)\; p(x_i|h)\; p(h)}{p(t)}. \end{aligned}$$

For a training instance $\langle x_i, y_i\rangle$, the quantity $p(y_i|x_i, h)$ measures the probability that h assigns the label y_i to the input feature vector x_i. The larger $p(y_i|x_i, h)$ is, the more likely h is to assign the correct label to x_i, and the smaller it is, the less likely h is to produce the correct label for x_i. Hence, we obtain the following definition of instance hardness, with respect to h:

$$\mathit{IH}_h(\langle x_i, y_i\rangle) = 1 - p(y_i|x_i, h).$$

In practice, h is induced by a learning algorithm g trained on t with hyper-parameters α, i.e., $h = g(t, \alpha)$. Explicitly, instance hardness equals $1-p(y_i|x_i, t, h)$ but since y_i is conditionally independent of t given h we can use $p(y_i|x_i, h)$. Thus, the hardness of an instance is dependent on the instances in the training data and the algorithm used to produce h. There are many approaches that could be taken to calculate instance hardness (or equivalently $p(y_i|x_i, g(t, \alpha)))$ such as an analysis of the distribution of instances in t according to their class. To gain a better understanding of what causes instance hardness in general, the dependence of instance hardness on a specific hypothesis can be lessened by summing instance hardness over the set of hypotheses $\mathcal{H}$ and weighting each $h \in \mathcal{H}$ by $p(h|t)$:

$$\begin{aligned} \mathit{IH}(\langle x_i, y_i\rangle) &= \sum_{\mathcal{H}} (1 - p(y_i|x_i, h))p(h|t) \\ &= \sum_{\mathcal{H}} p(h|t) - \sum_{\mathcal{H}} p(y_i|x_i, h)p(h|t) \\ &= 1 - \sum_{\mathcal{H}} p(y_i|x_i, h)p(h|t). \end{aligned} \tag{1}$$

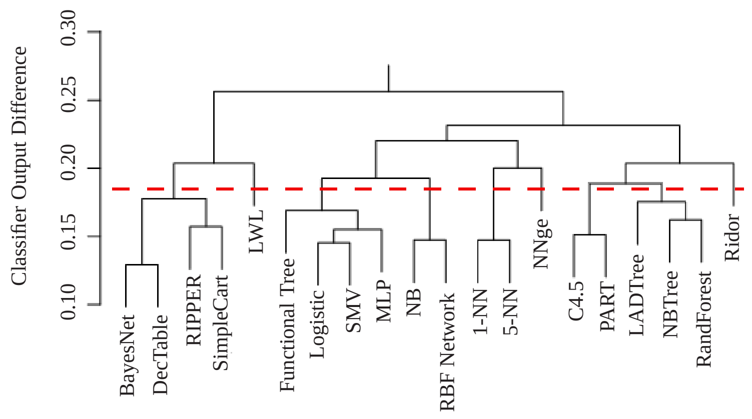

Fig. 1 Dendrogram of the considered learning algorithms clustered using unsupervised metalearning.

Practically, to sum over $\mathcal{H}$, one would have to sum over the complete set of hypotheses, or, since $h = g(t, \alpha)$, over the complete set of learning algorithms and hyper-parameters associated with each algorithm. This, of course, is not feasible. In practice, instance hardness can be estimated by restricting attention to a carefully chosen set of representative algorithms (and parameters). Also, it is important to estimate $p(h|t)$ because if all hypotheses were equally likely, then all instances would have the same instance hardness value under the no free lunch theorem (Wolpert, 1996). A natural way to approximate the unknown distribution $p(h|t)$, or equivalently $p(g(t, \alpha))$, is to weigh a set of representative learning algorithms, and their associated parameters, $\mathcal{L}$, a priori with a non-zero probability while treating all other learning algorithms as having zero probability. Given such a set $\mathcal{L}$ of learning algorithms, we can then approximate Equation 1 to the following:

$$IH_{\mathcal{L}}(\langle x_i, y_i\rangle) = 1 - \frac{1}{|\mathcal{L}|}\sum_{j=1}^{|\mathcal{L}|} p(y_i|x_i, g_j(t, \alpha)) \qquad (2)$$

where $p(h|t)$ is approximated as $\frac{1}{|\mathcal{L}|}$ and the distribution $p(y_i|x_i, g_j(t, \alpha))$ is estimated using the indicator function and classifier scores, as described in Section 4. For simplicity, we refer to *IH*$_{\mathcal{L}}$ as simply *IH* proceeding forward.

In this paper, we estimate instance hardness by biasing the selection of representative learning algorithms to those that 1) have shown utility, and 2) are widely used in practice. We call such classification learning algorithms the *empirically successful learning algorithms* (ESLAs). To get a good representation of $\mathcal{H}$, and hence a reasonable estimate of *IH*, we select a diverse set of ESLAs using unsupervised metalearning (Lee and Giraud-Carrier, 2011). Unsupervised metalearning uses Classifier Output Difference (COD) (Peterson and Martinez, 2005) to measure the diversity between learning algorithms. COD measures the distance between two learning algorithms as the probability that the learning algorithms make different predictions. Unsupervised metalearning then clusters the learning algorithms based on their COD scores with hierarchical agglomerative clustering. Here, we considered 20 commonly used learning algorithms with their default parameters as set in Weka (Hall et al, 2009). The resulting dendrogram is shown in Figure 1, where the height of the line connecting two clusters corresponds to the distance (COD value) between them. A cut-point

Table 1 Set $\mathcal{L}$ of ESLAs used to calculate instance hardness.

Learning Algorithms	
* RIpple DOwn Rule learner (RIDOR)	* Naïve Bayes
* Multilayer Perceptron trained with Back Propagation	* Random Forrest
* Locally Weighted Learning (LWL)	* 5-nearest neighbors (5-NN)
* Nearest Neighbor with generalization (NNge)	* Decision Tree (C4.5 (Quinlan, 1993))
* Repeated Incremental Pruning to Produce Error Reduction (RIPPER)	

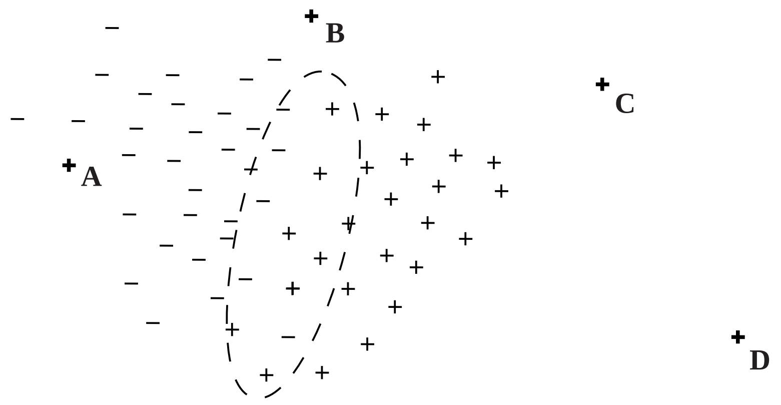

Fig. 2 Hypothetical 2-dimensional data set.

of 0.18 was chosen and a representative algorithm from each cluster was used to create $\mathcal{L}$ as shown in Table 1.

We recognize that instance hardness could be calculated with either more specific or broader sets of learning algorithms, and each set would obtain somewhat different results. We also recognize that the set of ESLAs is constantly evolving and thus no exact solution is possible. As the set of ESLAs grows and evolves, instance hardness can follow this evolution by simply adjusting $\mathcal{L}$. The size and exact make up of $\mathcal{L}$ are not as critical as getting a fairly representative sample of ESLAs. While more learning algorithms may give a more accurate estimate of instance hardness, we demonstrate that both efficiency and accuracy can be achieved with a relatively small and diverse set of learning algorithms.

With this approach, the instance hardness of an instance is dependent both on the learning algorithm trying to classify it and on its relationship to the other instances in the data set as demonstrated in the hypothetical two-dimensional data set shown in Figure 2. Instances A, C, and D could be considered outliers, though they vary in how hard they are to classify correctly: instance A would almost always be misclassified while instances C and D would almost always be correctly classified. The instances inside of the dashed oval represent *border points*, which would have a greater degree of hardness than the non-outlier instances that lie outside the dashed oval. Obviously, some instances are harder for some learning algorithms than for others. For example, some instances (such as instance B) are harder for a linear classifier than for a non-linear classifier because a non-linear classifier is capable of producing more complex decision boundaries.

3 Hardness Measures

In this section, we present a set of measures that measure various aspects about the instance hardness level of an individual instance. Instance hardness indicates which instances *are* misclassified while the hardness measures are intended to indicate *why* they are misclassified. Each hardness measure measures an aspect of why an instance may be misclassified (class overlap, class skew, etc.) and, thus, gives key insights into: 1) why particular instances are hard to classify, 2) how we could detect them, and 3) potentially creating improved mechanisms to deal with them. In addition, a subset of the measures could be used as a less expensive alternative to estimate instance hardness, although this is not investigated in this paper.

The set of hardness measures was discovered by examining the learning mechanisms of several learning algorithms. In compiling a set of hardness measures, we chose to use those that are relatively fast to compute and are interpretable so as to provide an indication as to why an instance is misclassified.

***k*-Disagreeing Neighbors (*k*DN).** *k*DN measures the local overlap of an instance in the original task space in relation to its nearest neighbors. The *k*DN of an instance is the percentage of the k nearest neighbors (using Euclidean distance) for an instance that do not share its target class value.

$$kDN(x) = \frac{|\ \{y : y \in kNN(x) \wedge t(y) \neq t(x)\}\ |}{k}$$

where $kNN(x)$ is the set of k nearest neighbors of x and $t(x)$ is the target class for x.

Disjunct Size (DS). DS measures how tightly a learning algorithm has to divide the task space to correctly classify an instance and the complexity of the decision boundary. Some learning algorithms, such as decision trees and rule-based learning algorithms, can express the learned concept as a disjunctive description. Thus, the DS of an instance is the number of instances in a disjunct divided by the number of instances covered by the largest disjunct in a data set.

$$DS(x) = \frac{|\ disjunct(x)\ | -1}{\max_{y \in D} |\ disjunct(y)\ | -1}$$

where the function $disjunct(x)$ returns the disjunct that covers instance x, and D is the data set that contains instance x. The disjuncts are formed using a slightly modified[1] C4.5 (Quinlan, 1993) decision tree, created without pruning and setting the minimum number of instances per leaf node to 1[2].

Disjunct Class Percentage (DCP). DCP measures the overlap of an instance on a subset of the features. Using a pruned C4.5 tree, the DCP of an instance is the number of

[1] The C4.5 algorithm stops splitting the data when a sufficient increase in information gain is not achieved. The idea of the DS measure is to overfit the data. However, not splitting all the way down led to impure disjuncts. Therefore, we modified the strictness of when to stop splitting such that all instances were carried out as far as they could go. The only impure disjuncts that remained are those for instances that have the same attribute values but differ in the class value.

[2] Note that C4.5 will create fractional instances in a disjunct for instances with unknown attribute values, possibly leading to DS values less than 1. Such cases are treated as though the disjunct covered a single instance.

instances in a disjunct belonging to its class divided by the total number of instances in the disjunct.

$$DCP(x) = \frac{\mid \{z : z \in disjunct(x) \wedge t(z) = t(x)\} \mid}{\mid disjunct(x) \mid}$$

Tree Depth (TD). Decision trees also provide a way to estimate the description length, or Kolmogorov complexity, of an instance. The depth of the leaf node that classifies an instance can give an intuition of the description length required for an instance. For example, an instance that requires 15 attribute splits before arriving at a leaf node is more complex than an instance that only requires 1 attribute split. Therefore, *tree depth* measures the depth of the leaf node for an instance in an induced C4.5 decision tree (both pruned (TD_P) and unpruned (TD_U)) as an estimate of the minimum description length for an instance.

Class Likelihood (CL). CL provides a global measure of overlap and the likelihood of an instance belonging to a class. The CL of an instance belonging to a certain class is defined as:

$$CL(x) = \prod_{i}^{|x|} P(x_i|t(x))$$

where $|x|$ is the number of attributes of instance x and x_i is the value of instance x's ith attribute[3]. The prior term is excluded in order to avoid bias against instances that belong to minority classes. CL assumes independence between the data attributes.

Class Likelihood Difference (CLD). CLD captures the difference in likelihoods and global overlap. It is the difference between the class likelihood of an instance and the maximum likelihood for all of the other classes.

$$CLD(x) = CL(x) - \underset{y \in Y - t(x)}{\text{argmax}}\ CL(x, y)$$

where Y represents set of possible labels in the data set.

Minority Value (MV). MV measures the skewness of the class that an instance belongs to. For each instance, its MV is the ratio of the number of instances sharing its target class value to the number of instances in the majority class.

$$MV(x) = 1 - \frac{\mid \{z : z \in D \wedge t(z) = t(x)\} \mid}{\max_{y \in Y} \mid \{z : z \in D \wedge t(z) = y\} \mid}.$$

Class Balance (CB). CB also measures the skewness of the class that an instance belongs to and offers an alternative to MV. If there is no class skew, then there is an equal number of instances for all classes. Hence, the CB of an instance is:

$$CB(x) = \frac{\mid \{z : z \in D \wedge t(z) = t(x)\} \mid}{\mid D \mid} - \frac{1}{\mid Y \mid}.$$

If the data set is completely balanced the class balance value will be 0.

3 Continuous variables are assigned a probability using a kernel density estimation (John, 1995).

Table 2 List of hardness measures and what they measure. The "+" and "-" symbols distinguish which hardness measures are positively and negatively correlated with instance hardness.

Abbr	+/-	Measure	Insight
*k*DN	+	*k*-Disagreeing Neighbors	Overlap of an instance using all of the data set features on a subset of the instances.
DS	-	Disjunct Size	Complexity of the decision boundary for an instance.
DCP	-	Disjunct Class Percentage	Overlap of an instance using a subset of the features and a subset of the instances.
TD	+	Tree Depth	The description length of an instance in an induced C4.5 decision tree.
CL	-	Class Likelihood	Overlap of an instance using all of the features and all of the instances.
CLD	-	Class Likelihood Difference	Relative overlap of an instance using all of the features and all of the instances.
MV	+	Minority Value	Class skew.
CB	-	Class Balance	Class skew.

For convenience, Table 2 summarizes the hardness measures and what they measure. Although all of the hardness measures are intended to understand why an instance is hard to classify, some of the measures indicate how easy an instance is to classify (they have a negative correlation with instance hardness). For example, the class likelihood (CL) measures how likely an instance belongs to a certain class. High values for CL would represent easier instances. In Table 2, the "+" and "-" symbols distinguish which hardness measures are positively and negatively correlated with instance hardness.

Class overlap and class skew are two commonly assumed and observed causes of instance hardness that are measured with the hardness measures. Mathematically, the class overlap of an instance for a binary task can be expressed as:

$$classOverlap(\langle x_i, y_i \rangle) = p(\bar{y}_i | x_i, t) - p(y_i | x_i, t). \tag{3}$$

where $\bar{y}_i$ represents an incorrect class for the input feature vector x_i. The class skew of the class of an instance can be expressed as:

$$classSkew(\langle x_i, y_i \rangle) = \frac{p(y_i | t)}{p(\bar{y}_i | t)}. \tag{4}$$

There is no known method to measure class overlap or to determine when class skew affects instance hardness. The hardness measures allow a user to estimate class overlap and class skew as well as other uncharacterized sources of hardness. Equations 3 and 4 could be extended to multi-class problems with a 1 vs. 1, or a 1 vs. all approach.

4 Experimental Methodology

In this section we provide our experimental methodology. Recall that to compute the instance hardness of an instance x, we must compute the probability that x is misclassified when the learner is trained on the other points from the dataset. Since this type of leave-one-out procedure is computationally prohibitive, the learning algorithms are evaluated using 5 by 10-fold cross-validation[4]. We use five repetitions to better measure the instance hardness

[4] 5 by 10-fold cross-validation runs 10-fold cross-validation 5 times, each time with a different random seed for selecting the 10 partitions of the data.

Table 3 Datasets used organized by number of instances, number of attributes, and attribute type.

# Instances	# Attributes	Attribute Type		
		Categorical	Numerical	Mixed
$M < 100$	$k < 10$	Balloons Contact Lenses		Post-Operative **cm1_req**
	$10 < k < 100$	Lung Cancer	**desharnais**	Labor Trains **Pasture**
$100 < M < 1000$	$k < 10$	Breast-w Breast Cancer	Iris Ecoli Pima Indians Glass Bupa Balance Scale	Badges 2 Teaching- Assistant
	$10 < k < 100$	Audiology Soybean(large) Lymphography Congressional- Voting Records Vowel Primary-Tumor Zoo	Ionosphere Wine Sonar Heart-Statlog **ar1**	Annealing Dermatology Credit-A Credit-G Horse Colic Heart-c Hepatitis Autos Heart-h **eucalyptus**
	$k > 100$		**AP_Breast_Uterus**	Arrhythmia
$1000 < M < 10000$	$k < 10$	Car Evaluation Chess Titanic	Yeast	Abalone
	$k < 100$	Mushroom Splice	Waveform-5000 Segment Spambase Ozone level- Detection	Thyroid- (sick & hypothyroid)
	$k > 100$		Musk (version 2)	
$M > 10000$	$k < 10$	Nursery	MAGIC Gamma- Telescope	
	$k < 100$	Chess- (King-Rook vs. King-Pawn) Letter		Adult-Census- Income (KDD) **Eye movements**

of each instance and to protect against the dependency on the data used in each fold. We then compare the hardness measures with instance hardness.

We examine instance hardness on a large and varied set of data sets chosen with the intent of being representative of those commonly encountered in machine learning problems. We analyze the instances from 57 UCI data sets (Frank and Asuncion, 2010) and 7 non-UCI data sets (Thomson and McQueen, 1996; Salojärvi et al, 2005; Sayyad Shirabad and Menzies, 2005; Stiglic and Kokol, 2009). Table 3 shows the data sets used in this study organized according to the number of instances, number of attributes, and attribute type. The non-UCI data sets are in bold.

We compare calculating instance hardness using all of the learning algorithms in $\mathcal{L}$ with calculating instance hardness using a single learning algorithm. In addition, $p(y_i|x_i, g(t,\alpha))$ is estimated using two methods: 1) the indicator function (IH_ind) which establishes the frequency of an instance being misclassified and 2) the classifier scores (IH_class). IH_ind and IH_class are calculated using 5 by 10-fold cross-validation. Generally, classification learning algorithms classify an instance into nominal classes. To produce a real-valued score, we calculate classifier scores for the nine investigated learning algorithms. Obviously, the indicator function and the classifier scores do not produce true probabilities. However, the classifier scores can provide the confidence of an inferred model for the class label of an instance. Below, we present how we calculate the classifier scores for the investigated learning algorithms.

Multilayer Perceptron: For multiple classes, each class from a data set is represented with an output node. The classifier score is the largest value of the output nodes normalized between zero and one (the *softmax* (Bridle, 1989)):

$$\hat{p}(y|x) = \frac{o_i(x)}{\sum_i^{|Y|} o_i(x)}$$

where y is a class from the set of possible classes Y and o_i is the value from the output node corresponding to class y_i

Decision Tree: To calculate a classifier score, an instance first follows the induced set of rules until it reaches a leaf node. The classifier score is number of training instances that have the same class as the examined instance divided by all of the training instances that also reach the same leaf node.

5-NN: 5-NN returns the percentage of the nearest-neighbors that agree with the class label of an instance as the classifier score.

LWL: LWL finds the *k*-nearest neighbors for an instance from the training data and weights them by their distance from the test instance. The weighted *k*-nearest neighbors are then used to train a base classifier. Weka uses a decision stump as the base classifier. A decision stump is a decision tree that makes a singe binary split on the most informative attribute. A test instance is propagated to a leaf node. The sum of weights of the training instances in the leaf node that have the same class value as the test instance is divided by the sum of the weights of all of the training instances in the leaf node.

Naïve Bayes: Returns the probability of the most probable class by multiplying the probability of the class by the probabilities of the attribute values for an instance given the class:

$$\max_{y_j \in Y} p(y_j) \prod_i^{|x|} p(x_i|y_j)$$

.

NNge: Since NNge only keeps exemplars of the training data, a class score of 1 is returned if an instance agrees with the class of the nearest exemplar, otherwise a 0 is returned.

Random Forest: Random forests return the class counts from the leaf nodes of each tree in the forest. The counts for each class are summed together and then normalized between 0 and 1.

RIDOR: RIDOR creates a set of rules, but does not keep track of the number of training instances covered by a rule. A classifier score of 1 is returned if RIDOR predicts the correct class for an instance, otherwise a 0 is returned (same as the indicator function).

RIPPER: RIPPER returns the percentage of training instances that are covered by a rule and share the same class as the examined instance.

To our knowledge, instance hardness is the only measurement that seeks to identify instances that are hard to classify. However, there are other methods that could be used to identify hard instances that have not been examined for identifying hard instances. One such method that we compare against is active learning. Active learning is a semi-supervised technique that uses a mode inferred from the labeled instances to choose which unlabeled instances are the most informative to be labeled by an external oracle. The informative scores assigned by active learning techniques can be used as a hardness measure. This assumes that the most informative instances are those that the model is least certain about, which would include the border points. We implemented two active learning techniques: uncertainty sampling (US) (Lewis and Gale, 1994) and query-by-committee (QBC) (Seung et al, 1992). For uncertainty sampling, we use margin sampling (Scheffer et al, 2001):

$$x^* = \underset{x}{\operatorname{argmin}}\, p(\hat{y}_1|x) - p(\hat{y}_2|x)$$

where $\hat{y}_1$ and $\hat{y}_2$ are the first and second most probable class labels for the instance x. We use naïve Bayes to calculate the probability of the classes for an instance. For query-by-committee, we use a committee of five learning algorithms using query by bagging (Abe and Mamitsuka, 1998). The level of disagreement is determined using vote entropy (Dagan and Engelson, 1995):

$$x^* = \underset{x}{\operatorname{argmax}} - \sum_i \frac{V(y_i)}{C} log \frac{V(y_i)}{C}$$

where y_i ranges over all possible class labels, $V(y_i)$ is the number of votes that a class label received from the committee, and C is the size of the committee. We examine QBC using naïve Bayes and decision trees. Active learning requires that some labeled instances are available to the models to produce the scores for the other instances. We divide the data set in half, using one half of the instances to calculate the scores for the other half.

We emphasize the extensiveness of our analysis. We examine over 190,000 instances individually. A total of 28,750 models are produced from 9 learning algorithms trained with 64 data sets using 5 by 10-fold cross-validation [5]. With this volume and diversity, our results can provide more useful insight about the extent to which hard instances exist and what contributes to instance hardness.

5 Instance-level Analysis

In this section we examine the hardness measures to identify hard instances and the hardness measures to discover what causes an instance to be misclassified. We use instance hardness with the indicator function (IH_ind) to establish the frequency of an instance being misclassified. Figure 3 shows the cumulative percentage of instances that are misclassified a specified percentage of times by the learning algorithms in $\mathcal{L}$ (Table 1). The first pair of columns shows that all of the instances were classified correctly by zero or more of the considered learning algorithms. The second pair of columns shows the percentage of instances that were misclassified by at least one of the considered learning algorithms. Overall, 2.4% of the instances from the UCI data sets are misclassified by all of the considered learning algorithms and 16.8% are misclassified by at least half. For the instances from the non-UCI data sets, 1.7% are misclassified by all of the considered learning algorithms and 22.7% are misclassified by at least half. The trend of hardness is similar for the UCI and non-UCI data sets. For the set of instances from the UCI and non-UCI data sets, only 38.3% of the instances are classified correctly 100% of the time by the examined learning algorithms.

[5] Ridor was not used on the Letter data set as it ran out of memory with 4 Gb of RAM. The remaining 8 learning algorithms still give a good indication of how difficult each instance is to correctly classify.

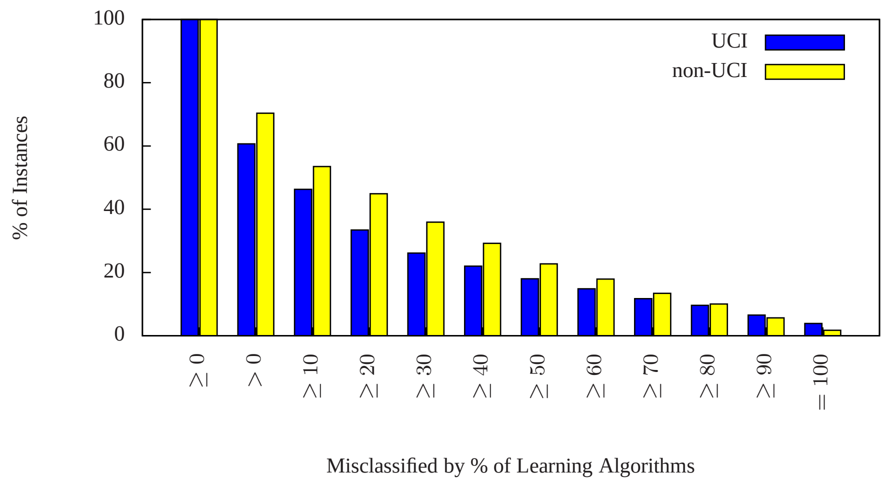

Fig. 3 Percentage of instances that are misclassified by at least a percentage of the learning algorithms.

Table 4 Spearman correlation matrix for the hardness measures. The magnitude of only one pair of measures is stronger than 0.95, showing that the measures measure different aspects of instance hardness.

	*k*DN	DS	DCP	TD_P	TD_U	CL	CLD	MV	CB
*k*DN	1.0	-0.519	-0.420	0.189	0.301	-0.715	-0.703	0.387	0.240
DS		1.0	0.570	-0.405	-0.348	0.571	0.559	-0.303	-0.139
DCP			1.0	-0.340	-0.202	0.452	0.432	-0.235	-0.051
TD_P				1.0	0.859	-0.276	-0.312	0.030	0.113
TD_U					1.0	-0.414	-0.441	0.162	0.293
CL						1.0	**0.989**	-0.386	-0.225
CLD							1.0	-0.359	-0.224
MV								1.0	0.783
CB									1.0

These results show that a considerable amount of instances are hard to classify correctly. Seeking to improve our understanding of why these instances are misclassified is the goal of the hardness measures.

We calculate the hardness measures for all of the instances regardless of their instance hardness. We first examine the relationship between the hardness measures. This will provide insight into how similar the measures are with each other and detect possible overlap in what they measure (see Table 2 for the hardness measures and what they measure). Next, we examine the relationship of the hardness measures with instance hardness. We first normalize the measures by subtracting the mean and dividing by the standard deviation for each measure before analyzing the results.

We first examine the correlation between the hardness measures. Table 4 shows a pairwise comparison of the hardness measures using the Spearman correlation. Only (CL) and class likelihood difference (CLD) are strongly correlated with a correlation coefficient of 0.989. This suggests that, besides CL and CLD, the hardness measures measure different properties of the hardness of an instance.

Table 5 The Spearman correlation coefficients for the hardness measures relating to the examined methods for identifying hard instances. The correlation coefficients with the strongest correlation with IH_ind and IH_class are in bold.

		*k*DN	DS	DCP	TD_P	TD_U	CL	CLD	MV	CB	Lin
Indicator Function	IH_ind	**0.830**	-0.547	-0.475	0.324	0.475	-0.670	-0.660	0.522	0.436	0.885
	MLP	**0.668**	-0.420	-0.397	0.270	0.354	-0.484	-0.476	0.367	0.287	0.725
	C.5	**0.625**	-0.459	-0.453	0.262	0.353	-0.469	-0.458	0.361	0.299	0.801
	5-NN	**0.648**	-0.307	-0.295	0.254	0.304	-0.319	-0.318	0.293	0.230	0.738
	LWL	**0.549**	-0.363	-0.302	0.149	0.288	-0.484	-0.470	0.506	0.447	0.671
	NB	**0.545**	-0.339	-0.302	0.162	0.284	-0.506	-0.499	0.405	0.328	0.626
	NNge	**0.716**	-0.464	-0.432	0.304	0.394	-0.552	-0.543	0.359	0.277	0.753
	RandFor	**0.669**	-0.490	-0.448	0.276	0.349	-0.487	-0.479	0.393	0.312	0.760
	Ridor	**0.711**	-0.437	-0.391	0.216	0.355	-0.575	-0.558	0.473	0.398	0.761
	RIPPER	**0.675**	-0.420	-0.387	0.172	0.306	-0.555	-0.538	0.496	0.350	0.747
Classifier Score	IH_class	**0.875**	-0.615	-0.540	0.341	0.513	-0.782	-0.767	0.542	0.425	0.938
	MLP	**0.764**	-0.515	-0.528	0.399	0.516	-0.679	-0.667	0.419	0.324	0.809
	C4.5	0.680	-0.629	**-0.711**	0.452	0.483	-0.644	-0.627	0.391	0.234	0.874
	5-NN	***0.771***	-0.316	-0.309	0.000	0.187	-0.574	-0.556	0.384	0.232	0.818
	LWL	**0.736**	-0.538	-0.432	0.198	0.426	**-0.745**	-0.718	0.615	0.521	0.874
	NB	0.698	-0.471	-0.411	0.205	0.382	**-0.775**	**-0.762**	0.411	0.273	0.779
	NNge	**0.716**	-0.464	-0.432	0.304	0.394	-0.552	-0.543	0.359	0.277	0.753
	RandFor	**0.852**	-0.611	-0.539	0.322	0.463	-0.724	-0.708	0.475	0.343	0.901
	Ridor	**0.711**	-0.437	-0.391	0.216	0.355	-0.575	-0.558	0.473	0.398	0.761
	RIPPER	**0.717**	-0.542	-0.583	0.417	0.486	-0.673	-0.649	0.398	0.263	0.854
	US_NB	-0.656	0.451	0.374	-0.097	-0.310	**0.889**	**0.881**	-0.373	-0.187	0.859
	QBC_NB	0.440	-0.229	-0.169	0.083	0.222	**-0.521**	**-0.534**	0.265	0.201	0.500
	QBC_C4.5	**0.672**	-0.486	-0.358	0.362	0.542	-0.605	-0.597	0.357	0.322	0.726

The more interesting question to consider is how does instance hardness relate to the considered hardness measures. Table 5 shows the Spearman correlation coefficients relating instance hardness to the other considered hardness measures for the UCI and non-UCI data sets. The hardness measure with the strongest correlation with instance hardness IH_ind and IH_class is in bold. The first section of the table uses the indicator function to calculate instance hardness, the second section uses the classifier scores to calculate instance hardness, and the third section shows the results for active learning. IH_ind and IH_class use the indicator function and classifier scores respectively from all of the learning algorithms in $\mathcal{L}$ to calculate instance hardness. The following rows use a single learning algorithm to calculate instance hardness. For all of the hardness measures, *k*DN, DCP, CL, and CLD have the strongest correlation with all of the hardness measures. Using PCA on the hardness measures, *k*DN, CL, and CLD have the largest coefficients for the first principal component (thus accounting for more variance than the other measures). *k*DN, CL, and CLD measure class overlap using all of the features from the data set. The other measures (which measure overlap on a subset of the features, class skew, and the description length) are not as indicative of an instance being hard to classify. The results from the Spearman correlation coefficients and the PCA analysis suggest that, in general, class overlap is a principal contributor to instance hardness for the considered data sets whether considering ESLAs in general (IH_ind and IH_class) or for a specific learning algorithm. The effect of class overlap on instance hardness can also be seen by examining individual instances and their corresponding hardness measures. The hardness measures and instance hardness values for a sample of instances are provided in Table 6. The first instance is a clear example that exhibits class overlap and should be misclassified as indicated by the values of the hardness measures (i.e. high value for *k*DN, CLD is negative, etc.).

Table 6 The hardness measures and instance hardness values for an example set of instances.

#	data	id	*k*DN	DS	DCP	TDP	TDU	CL	CLD	MV	CB	IH_ind	IH_cla	US_NB	QBC_NB	QBC_C4.5
1	yeast	1470	0.98	0	0.25	12	9	0.28	-0.32	0.47	0.06	1	0.84	0.20	0.09	0.60
2	colon	56	0.46	1	0.98	4	3	1	1	0	0.15	0.84	0.69	1	0	1
3	ar1	84	0.37	0	0.33	5	2	0.92	0.84	0	0.59	0.91	0.71	1	0.74	0.55

One of the difficulties of identifying hard instances is that hardness may arise from several sources. For example, instances 2 and 3 in Table 6 have multiple possible reasons for why they are misclassified, but no hardness measure strongly indicates that it should be misclassified (i.e. the *k*DN values are less than 0.5, meaning that the instances agree with the majority of their neighbors). The last column "Lin" in Table 5 shows the correlation coefficients of a linear model of the hardness measures predicting the hardness measures. The instance hardness and hardness measures from the UCI and non-UCI data sets for each instance were compiled and linear regression was used to predict the hardness measures. Apart from US_NB and QBC_NB, a linear combination of the hardness measures results in a stronger correlation with instance hardness than any of the individual measures suggesting that there is no one measure that sufficiently captures the hardness of an instance.

Comparing the hardness measures, IH_class has the strongest correlation with the linear combination of the hardness measures and *k*DN. IH_class also has a strong correlation with CL and CLD. Only US_NB has a stronger correlation with CL and CLD than IH_class. These strong correlations with IH_class suggest that IH_class may be a good candidate for determining the hardness of an instance. The QBC methods are not as strongly correlated with any of the hardness measures as the other hardness measures. The active learning approaches select border points as the hardest instances, but do not indicate that the outlier instances are hard. We also observe that using classifier scores has a stronger correlation with the hardness measures than using the indicator function to calculate instance hardness. For all of the considered learning algorithms, calculating instance hardness with the classifier scores provide a stronger or equal correlation with the hardness measures than the indicator function, suggesting that the classifier scores may provide a better indication of which instances are hard to classify. Also, for our examination of when ESLAs misclassify an instance, using an ensemble of learning algorithms to determine hardness has a stronger correlation with the hardness measures than a single learning algorithm.

The previous results suggest that, in general, class overlap causes instance hardness. However, in making this point, we realize that all data sets have different levels and causes of hardness. Table 7 shows the correlation between IH_class and the hardness measures for the instances in each data set. The column "DSH" refers to the *data set hardness* and is the average IH_class value for the instances in the data set. The harder data sets have a higher DSH value. The values in bold represent the hardness measures that have the strongest correlation with IH_class for the instances in the data set. The underlined values are the hardness measures with a correlation magnitude greater than 0.75. The values in Table 7 indicate that the hardness of the majority of the data sets is strongly correlated with the hardness measures that measure class overlap. There are a few data sets that have a strong correlation between IH_class and the measures that measure class skew (MV and CB). The most notable are the post-opPatient and zoo data sets. For those data sets, in addition to having a strong correlation with MV and CB, instance hardness is also strongly correlated with other hardness measures that measure class overlap.

It is not surprising that class overlap is observed as a principal contributor to instance hardness since outliers and border points, which exhibit class overlap, have been observed

Table 7 The correlation of the hardness measures with IH_class for the instances in each data set. DSH is the average IH_class value of the instances in the data set. The underlined values are the hardness measures with a correlation magnitude greater than 0.75. The bold values represent the hardness measures that have the strongest correlation with IH_class for each data set.

Dataset	DSH	*k*DN	DS	DCP	TD_P	TD_U	CL	CLD	MV	CB
abalone	0.815	**0.859**	-0.485	-0.287	-0.203	-0.085	-0.194	-0.141	0.323	0.323
adult-census	0.208	**0.898**	-0.722	-0.743	0.515	0.599	-0.737	-0.737	0.569	0.569
anneal.ORIG	0.108	0.658	-0.600	-0.349	0.326	0.416	**-0.689**	-0.687	0.425	0.425
AP_BreastUterus	0.056	0.563	**-0.615**	-0.279	0.094	0.323	-0.168	-0.168	0.535	0.535
ar1	0.126	0.684	**-0.814**	-0.388	0.726	0.395	0.450	0.450	0.450	0.450
arrhythmia	0.416	**0.845**	-0.769	-0.334	-0.655	-0.478	-0.404	-0.407	0.687	0.687
audiology	0.339	**0.836**	-0.783	-0.262	-0.011	-0.010	-0.681	-0.668	0.653	0.653
autos	0.337	**0.752**	-0.405	-0.082	0.064	0.033	-0.450	-0.447	0.086	0.086
badges2	0.003	0.216	**-0.563**	NA	NA	NA	-0.377	-0.377	**0.563**	**0.563**
balance-scale	0.259	**0.935**	-0.851	-0.578	0.792	0.749	-0.775	-0.797	0.466	0.466
balloons	0.072	0.746	-0.390	NA	0.035	0.035	**-0.931**	**-0.931**	-0.283	-0.283
breast-cancer	0.339	**0.877**	-0.632	-0.764	0.256	0.265	-0.490	-0.490	0.645	0.645
breast-w	0.059	0.627	**-0.809**	-0.610	0.746	0.804	-0.525	-0.533	0.598	0.598
bupa	0.396	**0.715**	-0.512	-0.358	0.395	0.404	0.191	0.191	0.389	0.389
carEval	0.140	0.924	-0.883	-0.561	0.886	0.873	**-0.937**	-0.912	0.705	0.705
chess	0.614	**0.606**	-0.245	-0.498	0.226	0.073	-0.310	-0.242	0.190	0.190
chess-KRVKP	0.087	0.608	-0.348	-0.317	0.725	0.726	**-0.860**	**-0.860**	0.126	0.126
cm1_req	0.324	0.628	-0.166	**-0.710**	0.548	NA	0.236	0.236	**0.710**	**0.710**
colic	0.223	**0.796**	-0.660	-0.644	0.325	0.296	-0.444	-0.443	0.282	0.282
colon	0.286	**0.620**	-0.528	0.391	-0.316	-0.342	-0.173	-0.173	0.495	0.495
contact-lenses	0.281	0.859	-0.880	-0.744	**0.907**	0.868	-0.877	-0.871	0.551	0.551
credit-a	0.197	**0.755**	-0.581	-0.743	0.367	0.574	-0.511	-0.511	0.360	0.360
credit-g	0.321	**0.887**	-0.595	-0.420	0.288	0.556	-0.620	-0.620	0.675	0.675
dermatology	0.099	**0.757**	-0.689	-0.444	0.494	0.457	-0.738	-0.744	0.526	0.526
desharnais	0.386	**0.877**	-0.714	-0.199	-0.024	-0.381	0.053	0.136	0.562	0.562
ecoli	0.229	**0.870**	-0.741	-0.234	-0.137	0.213	-0.831	-0.829	0.712	0.712
eucalyptus	0.467	**0.845**	-0.632	-0.506	0.380	0.352	-0.390	-0.381	0.298	0.298
eye_movements	0.492	**0.618**	-0.459	-0.115	0.198	0.254	-0.289	-0.265	-0.159	-0.159
glass	0.399	**0.816**	-0.606	0.045	0.136	0.144	-0.477	-0.474	-0.055	-0.055
heart-c	0.244	**0.816**	-0.756	-0.314	0.413	0.368	-0.740	-0.743	0.046	0.046
heart-h	0.237	**0.803**	-0.751	-0.563	0.297	-0.181	-0.675	-0.681	0.394	0.394
heart-statlog	0.248	**0.793**	-0.672	-0.187	0.496	0.497	-0.719	-0.719	0.095	0.095
hepatitis	0.222	0.825	**-0.888**	-0.011	0.550	0.635	-0.715	-0.715	0.615	0.615
hypothyroid	0.039	**0.655**	-0.281	-0.144	0.284	0.272	-0.617	-0.619	0.449	0.449
ionosphere	0.138	0.350	**-0.556**	0.044	0.182	0.183	-0.265	-0.262	0.119	0.119
iris	0.071	0.598	-0.579	-0.463	**0.870**	0.846	-0.626	-0.626	NA	NA
labor	0.177	**0.667**	-0.455	-0.514	0.336	-0.248	-0.553	-0.553	0.389	0.389
letter	0.347	0.752	**-0.790**	-0.244	0.518	0.588	-0.785	-0.769	0.040	0.040
lungCancer	0.537	**0.736**	-0.393	-0.251	0.283	0.203	-0.052	-0.057	-0.200	-0.200
lymphography	0.251	**0.776**	-0.759	-0.113	0.291	0.299	-0.546	-0.538	0.246	0.246
MagicTelescope	0.223	**0.821**	-0.624	-0.387	-0.116	0.088	-0.611	-0.612	0.435	0.435
mushroom	0.016	0.085	-0.342	NA	**-0.565**	**-0.565**	-0.147	-0.140	0.451	0.451
nursery	0.110	0.569	-0.879	-0.384	0.896	**0.904**	-0.897	-0.892	0.717	0.717
ozone	0.071	0.550	-0.547	-0.217	0.224	**0.587**	-0.499	-0.503	0.290	0.290
pasture	0.295	**0.745**	-0.672	-0.133	0.569	0.667	-0.671	-0.673	NA	NA
pimaDiabetes	0.305	**0.895**	-0.696	-0.625	0.510	0.659	-0.622	-0.622	0.481	0.481
post-opPatient	0.425	0.785	-0.573	**-0.788**	0.145	NA	-0.079	-0.071	0.775	0.775
primary-tumor	0.678	**0.887**	-0.486	-0.754	0.260	0.311	-0.539	-0.489	0.476	0.476
segment	0.115	0.616	**-0.911**	-0.476	0.783	0.719	-0.636	-0.637	NA	NA
sick	0.037	0.591	0.002	0.407	0.137	0.331	**-0.772**	**-0.772**	0.390	0.390
sonar	0.274	0.652	-0.568	-0.330	0.303	0.264	**-0.715**	**-0.715**	0.027	0.027
soybean	0.181	**0.820**	-0.718	-0.221	0.275	0.259	-0.586	-0.591	0.138	0.138

Continued on next page

Table 7 (cont.) The correlation of the hardness measures with IH_class for the instances in each data set. DSH is the average IH_class value of the instances in the data set. The underlined values are the hardness measures with a correlation magnitude greater than 0.75. The bold values represent the hardness measures that have the strongest correlation with IH_class for each data set.

Dataset	DSH	*k*DN	DS	DCP	TD_P	TD_U	CL	CLD	MV	CB
spambase	0.133	0.583	**-0.655**	-0.273	0.343	0.261	-0.603	-0.616	0.037	0.037
splice	0.158	0.373	-0.549	-0.402	0.520	0.504	**-0.674**	-0.673	0.283	0.283
teachingAssistant	0.495	**0.790**	-0.629	-0.388	0.055	-0.161	-0.219	-0.218	0.066	0.066
titanic	0.305	0.356	0.272	**-0.888**	0.030	0.030	-0.140	-0.140	0.256	0.256
trains	0.411	**0.756**	-0.362	-0.241	0.071	NA	-0.200	-0.200	NA	NA
vote	0.070	0.674	**-0.825**	-0.622	0.811	0.691	-0.645	-0.654	0.549	0.549
vowel	0.287	0.364	**-0.521**	-0.190	0.343	0.320	-0.443	-0.403	NA	NA
waveform-5000	0.268	**0.805**	-0.595	0.116	0.419	0.417	-0.651	-0.651	-0.184	-0.184
wine	0.079	**0.711**	-0.519	-0.242	-0.093	-0.368	-0.538	-0.539	-0.102	-0.102
yeast	0.523	**0.893**	-0.674	-0.342	0.081	0.285	-0.285	-0.222	-0.026	-0.026
zoo	0.134	**0.900**	-0.836	-0.392	0.690	0.684	-0.828	-0.821	0.830	0.830

Table 8 Various statistics for the hardness measures for instances that belong to the majority class and those that do not. For the instances that belong to the minority class, the values for the measures indicate higher levels of class overlap. The column "easy" gives the expected value for the hardness measure if an instance has low instance hardness.

	Minority Class				Majority Class				
	Min.	Mean	Max.	Std Dev	Min.	Mean	Max.	Std Dev	easy
DN	0	0.348	1	0.307	0	0.172	1	0.236	0
DS	0	0.179	1	0.286	0	0.363	1	0.410	1
DCP	0	0.786	1	0.300	0.002	0.909	1	0.170	1
TD_P	1	9.530	59.88	6.232	1	9.593	136.265	7.355	1
TD_U	0	7.315	29	4.809	0	5.555	29	4.526	1
CL	0	0.563	1	0.375	0	0.886	1	0.188	1
CLD	-1	0.297	1	0.602	-1	0.803	1	0.306	1
MV	1	0.308	0.910	0.278	0	0	0	0	1
CB	-0.471	-0.026	0.189	0.111	0	0.213	0.213	0.155	1
IH_class	0	0.410	0.999	0.269	0	0.163	0.994	0.200	0

to be more difficult to classify correctly. However, instances that belong to a minority class have also been observed to be more difficult to classify correctly. This is confirmed as the coefficients for the class imbalance measures (MV and CB) in the linear regression models are statistically significant. Also, removing MV and CB from the linear model results in a weaker correlation. To what extent does class skew affect instance hardness? One of the core problems seen with class skew is that of data ambiguity, when multiple instances have the same feature values but different classes. In these cases, the instances that belong to the minority class will be misclassified. There are only 204 such instances, about 0.1% of all of the instances used in this study. We removed all of the ambiguous instances and then divided the instances into those that have a MV value of 0 (they belong to the majority class) and those that have a value greater than 0. This considers any instance that does not belong to the majority class as belonging to a minority class. There are 97,469 instances that belong to the majority class and 92,669 instances that do not. We observe that instances that belong to a minority class are harder to classify correctly than those that do not. The average IH_class value for the instances that belong to a majority class is 0.16 while the average instance hardness value for the instances not belonging to the majority class is 0.41. Table 8 compares the hardness measures for the instances that belong to a minority class and those that belong

Table 9 The hardness measures and instance hardness values for an example set of instances from the chess data set.

id	*k*DN	DS	DCP	TDP	TDU	CL	CLD	MV	CB	IH_ind	IH_cla	US_NB	QBC_NB	QBC_C4.5
22037	0.64	0.29	1	5	5	0.09	-0.36	0	0.11	0.24	0.41	0.33	0.00	0.40
26549	0.64	0.29	1	5	5	0.12	-0.36	0.52	0.02	0.49	0.39	0.39	0.00	0.32

to the majority class. The last column (easy) gives the value for the hardness measures for the easiest instances (the instances that are always correctly classified). Not including MV and CB (which are biased since all of the instances that belong to the non-majority classes are separated from the majority class instances), all of the hardness measures except for pruned tree depth (TD_P) indicate that the instances that belong to a minority class are harder to classify correctly as well. Thus, we observe that class skew exacerbates the effects of the underlying causes for instance hardness. This coincides with Batista's conclusion that class skew alone does not hinder learning algorithm performance, but rather class skew magnifies the hardness already present in the instances (Batista et al, 2004). For example, Table 9 gives the hardness measure values for two instances from the chess data set. The hardness measures are similar for each measure except the first instance (id 22037) belongs to the majority class while instance 26549 does not. The difference in the IH_ind value for the instances is considerable. The difference in IH_class values does not vary considerably since many of the class scores are similar to the hardness measures. This supports the fact that class skew exacerbates the effects of class overlap and also shows that IH_ind may be better able to incorporate the effects of class skew than IH_class. Given that class skew exacerbates the effects of class overlap on instance hardness, the expected instance hardness for an instance is related to the class overlap (Equation 3) and class skew (Equation 4) of the instance:

$$\mathbb{E}[IH(\langle x_i, y_i\rangle)] \sim f(classOverlap(\langle x_i, y_i\rangle),\ classSkew(\langle x_i, y_i\rangle)).$$

The exact form of f is unknown at this stage. Additionally, other factors not discussed here may affect the hardness of an instance. Discovering the relationship between class overlap, class skew, and instance hardness, as well as identifying other sources of hardness, is left for future work.

6 Integrating Instance Hardness into the Learning Process

In this section we examine how to exploit instance hardness during the learning process to alleviate the effects of class overlap and instance hardness. Incorporating instance hardness into the learning process provides significant improvements in accuracy. Note that the improvement requires computing instance hardness for each instance. In the experiments, we opt to use IH_class instead of IH_ind as they are strongly correlated and IH_class produces slightly better results. We also ran the experiments calculating instance hardness with the same single learning algorithm that is inferring the model. This provides the opportunity to compare whether it is more appropriate to use a specific measure of instance hardness rather than a more general one. In addition, we ran the experiments using the active learning hardness measures. The active learning techniques are not designed to identify hard instances and using them as a hardness measure often resulted in poor results. In order to avoid a deluge of data, we do not show their results.

6.1 Informative Error

Informative error (IE) is based on the premise of knowing if an instance *should* be misclassified. We implement IE in multilayer perceptrons (MLPs) trained with backpropagation using instance hardness computed using 1) all of the learning algorithms in $\mathcal{L}$ (IE$_{ESLA}$) and 2) only using a MLP (IE$_{MLP}$). We use instance hardness to estimate if an instance should be misclassified. A common approach for classification problems with MLPs is to create one output node for every class value. If the data set has a class with three possible values, then three output nodes are created. The target output value for each node is either 1 if the instance belongs to that class or 0 if it does not. The error function of $target - output$ for each of the k output nodes can then be formulated as:

$$error(x) = \begin{cases} 1 - o_k & \text{if } t(x) = k_{class} \\ 0 - o_k & \text{otherwise} \end{cases}$$

where o_k is the output value for node k, $t(x)$ is the target value for instance x and k_{class} is the class represented by node k.

We modify the error function such that it subtracts the instance hardness value of an instance from the target value for the output node only.

$$error(x) = \begin{cases} 1 - IH(x, t(x)) - o_k & \text{if } t(x) = k_{class} \\ 0 - o_k & \text{otherwise} \end{cases}$$

The instance hardness value is only subtracted from the output node that corresponds with the target class value of an instance. If the instance hardness value were added to the output value for the output nodes that do not correspond with the target class value of an instance then this could potentially confuse the network as an instance is incorrect for one class value yet correct for all of the others. For example, if an instance has an instance hardness value of 1, then the errors would essentially tell the network that the target value is wrong whereas all of the other classes are correct. Also, if an instance had an instance hardness value of 0.5, all output nodes would have the same target value and no information is gained. IE places more emphasis on the non-overlapping instances by reducing the weight of the error from instances with high instance hardness values.

Table 10 shows the results of using IE to train a MLP on 52 data sets (the data sets that did not have instance hardness greater than 0.5 were not used) compared against two filtering techniques (repeated edited nearest neighbor (RENN) (Tomek, 1976) and fast local kernel noise reduction (FaLKNR) (Segata et al, 2009)) and two boosting methods (AdaBoost (Freund and Schapire, 1996) and MultiBoost (Webb, 2000) using a MLP as the base algorithm). RENN repeatedly removes the instances that are misclassified by a 3-nearest neighbor classifier and has produced good results. FaLKNR removes any instances that disagree with the predicted class from a support vector machine trained on the neighborhood of the selected instance. The average accuracy, the number of times that the accuracy using IE$_{MLP}$ is better, the same, or worse than the other methods, and the *p*-value calculated using the Wilcoxon signed-rank test are provided in the bottom three rows as a summary of the table. There are 14 data sets on which IE$_{MLP}$ increases accuracy by more than 5%, indicated by an asterisk. On the lung cancer data set, accuracy increases by 21.9% and is 3 percentage points higher than the next best algorithm (FaLKNR). On the labor data set, IE$_{MLP}$ increases accuracy by 10.5% and is 5 percentage points greater than the next best algorithm. On average, IE$_{MLP}$ increases more than 3% in accuracy over the original and

Table 10 Pairwise comparison of informative error with standard backpropagation, RENN, FaLKNR, AdaBoost, and MultiBoost. An asterisk indicates data sets on which IE$_{MLP}$ improves accuracy more than 5%.

Dataset	Orig	RENN	FaLKNR	AdaBoost	MultiBoost	IE$_{ESLA}$	IE$_{MLP}$
abalone	26.24	27.80	28.78	26.24	26.24	27.84	**29.12**
adult-census	82.91	83.82	83.45	82.91	82.91	**85.22**	84.46
anneal.ORIG	98.78	98.33	97.55	98.89	98.89	**99.33**	96.82
arrhythmia	67.70	61.50	67.92	67.70	67.70	**71.68**	71.06
audiology	**83.19**	74.78	77.88	**83.19**	**83.19**	79.65	81.95
autos	80.00	76.10	68.29	79.51	78.05	78.05	**80.39**
balance-scale	90.72	89.45	90.72	92.64	**92.80**	90.40	91.71
breast-cancer*	64.69	74.48	73.08	70.28	69.93	**75.87**	74.34
breast-w	95.28	96.14	96.28	94.99	95.85	96.57	**96.62**
bupa	71.59	71.88	71.01	71.88	71.59	**72.75**	71.59
carEval	**99.54**	92.53	99.25	**99.54**	**99.54**	98.61	99.46
chess-KRvsKP	99.41	99.31	**99.47**	99.41	99.41	99.41	99.39
chess	62.25	56.94	62.18	65.75	**67.60**	51.49	61.31
colic*	80.43	83.70	85.33	80.98	82.07	85.33	**86.25**
contact-lenses*	70.83	**91.67**	83.33	70.83	70.83	**91.67**	87.50
credit-a	84.20	**87.97**	84.78	84.20	84.35	86.52	87.62
credit-g*	71.60	76.00	72.80	71.50	73.00	76.90	**78.22**
dermatology	96.17	96.45	97.81	96.17	96.17	97.54	**98.69**
ecoli	86.01	87.20	**87.50**	84.23	85.12	86.90	87.20
glass	67.76	70.09	68.22	71.50	67.29	70.56	**71.96**
heart-c*	80.86	82.51	79.54	77.56	79.87	84.16	**86.20**
heart-h	**85.03**	84.35	84.01	80.27	81.63	82.99	84.42
heart-statlog*	78.15	82.96	83.33	78.15	80.37	**85.93**	84.81
hepatitis*	80.00	86.45	82.58	79.35	79.35	87.10	**89.68**
hypothyroid	94.04	94.35	94.57	94.62	95.10	**95.52**	94.90
ionosphere	91.17	86.61	86.61	91.17	**91.74**	**91.74**	89.23
iris	**97.33**	96.67	96.67	96.67	96.67	96.67	96.80
labor*	85.96	85.97	91.23	85.96	85.96	**96.49**	91.93
letter	82.08	82.56	82.60	**88.35**	87.30	80.48	81.86
lungCancer*	37.50	50.00	56.25	37.50	37.50	59.38	**60.00**
lymphography	84.46	83.78	83.11	84.46	85.14	85.14	**85.95**
MagicTelescope	85.87	85.42	85.19	85.90	86.25	85.53	**86.36**
nursery	99.73	97.45	98.89	**99.97**	**99.97**	99.87	99.92
ozone	96.41	96.81	97.12	96.41	**99.73**	96.96	97.41
pimaDiabetes	75.39	76.69	75.91	75.26	75.13	77.08	**78.07**
post-opPatient*	55.56	71.11	71.11	52.22	54.44	66.67	**72.22**
primary-tumor*	38.35	47.20	46.02	43.07	43.07	49.56	**51.27**
segment	96.06	95.97	96.41	96.06	95.93	96.10	**96.84**
sick	97.27	97.27	96.85	97.27	97.11	**97.51**	97.42
sonar*	82.21	84.13	85.10	83.65	83.17	87.98	**88.17**
soybean	93.41	92.97	**95.17**	93.41	93.41	**95.17**	94.73
spambase	91.44	91.05	92.18	91.44	91.05	92.24	**92.65**
splice	95.96	95.24	95.36	95.96	95.96	**96.80**	96.78
teachingAssistant*	58.94	61.59	63.58	58.94	58.94	64.90	**65.56**
titanic	78.46	**79.06**	**79.06**	78.60	78.96	78.87	**79.06**
trains*	70.00	80.00	50.00	70.00	70.00	**90.00**	**90.00**
vote	94.71	94.71	**96.55**	94.48	94.48	95.17	95.95
vowel	92.73	93.84	93.64	96.26	**96.67**	91.62	91.94
waveform-5000	83.56	84.93	86.30	83.36	83.50	85.66	**86.60**
wine	97.19	96.63	96.63	97.19	97.19	97.75	**98.65**
yeast	59.43	59.03	60.31	59.43	59.10	59.97	**60.77**
zoo	**96.04**	94.06	94.06	**96.04**	**96.04**	**96.04**	95.84
Average	81.05	82.45	82.14	81.37	81.60	84.03	**84.57**
better-same-worse	40-1-11	43-0-9	43-1-8	40-0-12	38-1-13	36-1-15	
p-Value	< 0.001	< 0.001	< 0.001	< 0.001	< 0.001	0.003	

2% over RENN. The increases in accuracy are statistically significant. In this case, IE$_{MLP}$ is significantly better than IE$_{ESLA}$. Thus, in this case, using a specific bias from a learning algorithm is preferred. This is examined in more detail in the next section.

Although IE is described in the context of MLPs, it can also be applied to other learning algorithms that are incrementally updated based on an error value such as the class of non-closed form regression models (i.e., logistic regression and isotonic regression). Similar to informative error, instance hardness could be used to weight the instances prior to training a model. This weight could then be used in a number of learning algorithms such as nearest-neighbor or naïve Bayes algorithms.

6.2 Filtering the data set

A simple idea to handle hard instances and reduce overlap is to filter or remove them from a data set prior to training. The idea of filtering is to remove the instances that are suspected outliers or noise and thus increase class separation (Smith and Martinez, 2011). We use the IH_class values to determine which instances to filter from the data sets. We compare the results to those by RENN and the majority and consensus filters proposed by Brodley and Friedl (1999). The majority and consensus filters remove an instance if it is misclassified respectively by the majority of, or all, three learning algorithms (C4.5, IB1, and thermal linear machine (Brodley and Utgoff, 1995)). When using the instance hardness values, we use the classifier scores from the five folds of the nine learning algorithms as our ensemble and remove any instances with an IH_class value greater than a set threshold. We set the threshold at 0.5 (IH_0.5), 0.7 (IH_0.7) and 0.9 (IH_0.9). We also compare using each learning algorithm to filter the instances and as the learning algorithm (IH_LA). For example, IH_LA for MLP uses a MLP to identify which instances to filter prior to training a MLP. Each filtering technique was used on a set of 52 data sets evaluated using five by ten-fold cross-validation on the nine learning algorithms. Testing is done on all of the data, including the instances that were removed.

For the nine learning algorithms, Table 11 shows the average accuracy, pairwise comparison of the accuracies, and *p*-values from the Wilcoxon signed-rank statistical significance test comparing the filtering method to the original accuracy. Only the averages are displayed to avoid the overload of tables and much of the aggregate information is present in the pairwise comparison of the algorithms (number of times that a learning algorithm increases-stays the same-decreases the accuracy) and the *p*-value from the Wilcoxon signed rank significance test. Filtering significantly increases classification accuracy for most of the filtering techniques and learning algorithms. IH 0.7 achieves the greatest increase in accuracy, being slightly better than the majority filter. One of the advantages of using instance hardness is that various thresholds can be used to filter the instances. However, we note that there is not one filtering approach that is best for all learning algorithms and data sets (as indicated by the counts). For filtering, using the same learning algorithm to infer the model and to determine which instances to filter is only better than using all of the learning algorithms in $\mathcal{L}$ for C4.5 and 5-NN.

To examine the variability of each data set and learning algorithm combination, we examine an adaptive filtering approach that generates a set of learning algorithms to calculate instance hardness for a specific data set/learning algorithm combination. We call the set of learning algorithms used to calculate instance hardness a *filter set*. The adaptive approach discovers the filter set through a greedy search of $\mathcal{L}$. The adaptive approach iteratively adds a learning algorithm from $\mathcal{L}$ to a filter set by selecting the learning algorithm that produces

Table 11 The average accuracy values for the nine learning algorithms comparing filtering techniques against not filtering the data (Orig). “count” gives the number of times that a filtering algorithm improves, maintains, or reduces classification accuracy. On average, filtering the data sets significantly improves the classification accuracy. The *p*-values in bold represent the cases where filtering significantly increases the classification accuracy over not filtering. For each learning algorithm, the accuracy for the filtering technique that produces the highest accuracy is in bold.

Algorithm	Orig	IH_0.5	IH_0.7	IH_0.9	RENN	Majority	Consen	IH_LA
MLP	81.05	83.12	**83.58**	81.86	82.45	82.52	81.92	82.95
count		35-1-16	37-0-15	30-2-20	28-3-21	25-0-27	24-0-28	36-0-16
p-value		**0.001**	**< 0.001**	**0.025**	**0.047**	0.151	0.246	**0.002**
C4.5	80.11	80.23	81.46	80.53	80.51	81.48	81.11	**82.06**
count		32-0-20	41-2-9	25-2-25	25-4-23	32-3-17	36-3-13	38-2-12
p-value		0.054	**< 0.001**	0.226	0.122	**0.002**	**0.001**	**< 0.001**5
5-NN	79.03	81.42	82.14	80.21	82.28	81.62	81.05	**82.34**
count		39-1-12	38-3-11	37-4-11	39-2-11	36-5-11	32-9-11	41-2-9
p-value		**< 0.001**	**< 0.001**	**< 0.001**	**< 0.001**	**< 0.001**	**< 0.001**	**< 0.001**
LWL	69.36	71.05	69.65	69.80	69.75	**70.80**	70.32	67.69
count		32-11-9	26-12-14	23-13-16	24-14-14	33-8-11	22-18-12	21-13-18
p-value		**0.002**	0.127	0.103	0.091	**0.002**	**0.009**	0.634
NB	75.68	77.79	77.22	**76.50**	76.17	77.52	77.05	75.04
count		37-1-14	36-0-16	32-3-17	27-7-18	35-4-13	31-8-13	21-1-30
p-value		**< 0.001**	**< 0.001**	**0.008**	0.083	**0.001**	**< 0.001**	0.871
NNge	79.45	81.69	**82.16**	80.05	81.16	81.40	81.10	81.57
count		34-0-18	41-0-11	29-2-21	30-3-19	31-4-17	29-7-16	36-0-16
p-value		**< 0.001**	**< 0.001**	0.070	**0.040**	**0.006**	**0.003**	**0.001**
RandFor	81.59	82.52	**83.07**	81.83	82.44	82.37	81.97	82.80
count		28-3-21	36-0-16	29-1-22	26-6-20	24-5-23	25-8-19	27-4-21
p-value		**0.00**9	**0.001**	0.081	**0.045**	0.051	**0.026**	**0.031**
Ridor	78.09	**79.29**	79.22	78.45	78.16	78.87	78.94	78.65
count		36-3-13	36-2-14	25-1-26	27-2-23	34-3-15	29-7-16	32-3-17
p-value		**< 0.001**	**0.001**	0.173	0.419	**0.003**	**0.021**	**0.013**
RIPPER	77.83	79.21	79.16	78.44	77.52	**79.79**	78.89	78.83
count		37-1-14	38-0-14	32-1-19	26-4-22	36-1-15	32-7-13	37-2-13
p-value		**< 0.001**	**< 0.001**5	**0.019**	0.464	**0.001**	**0.001**	**0.003**
Average	78.02	79.59	**79.74**	78.63	78.94	79.60	79.15	79.10
count		42-0-10	45-0-7	38-0-14	33-1-18	38-0-14	39-0-13	38-0-14
p-value		**< 0.001**	**< 0.001**	**< 0.001**	**0.003**	**< 0.001**	**< 0.001**	**0.001**

the highest classification accuracy when added to the filter set, as shown in Algorithm 1. A constant threshold value is set to filter instances in $runLA(F)$ for all iterations. We examine thresholds of 0.5, 0.7, and 0.9. The baseline accuracy for the greedy approach is the accuracy of the learning algorithm without filtering. The search stops once adding one of the remaining learning algorithms to the filter set does not increase accuracy. The running time for the adaptive approach is $O(N^2)$ where N is the number of learning algorithms to search over. The significant improvement in accuracy makes the increase in computational time reasonable in most cases.

Table 12 gives the results for adaptively filtering for a specific data set/learning algorithm combination. The adaptive approach significantly increases the classification accuracy over IH_0.7 for all of the learning algorithms and thresholds. The accuracy increases for at least 85% of the data sets regardless of which learning algorithm is being used for classification. A_0.9 achieves the highest classification accuracy for the adaptive approach. Interestingly, there is no one particular learning algorithm that is always included in a filter set for a particular learning algorithm. The frequency for how often a learning algorithm is included

Algorithm 1 Adaptively constructing a filter set.

```
1: Let $F$ be the filter set used for filtering and $\mathcal{L}$ be the set of candidate learning algorithms for $F$.
2: Initialize $F$ to the empty set: $F \leftarrow \{\}$
3: Initialize the current accuracy to the accuracy from an empty filter set: $currAcc \leftarrow runLA(\{\})$.
   $runLA(F)$ returns the accuracy from a learning algorithm trained on a data set filtered with $F$.
4: while $\mathcal{L} \neq \{\}$ do
5:     $bestAcc \leftarrow currAcc$;
6:     $bestLA \leftarrow null$;
7:     for all $g \in \mathcal{L}$ do
8:         $tempF \leftarrow F + g$;
9:         $acc \leftarrow runLA(tempF)$;
10:        if $acc > bestAcc$ then
11:            $bestAcc \leftarrow acc$;
12:            $bestLA \leftarrow g$;
13:        end if
14:    end for
15:    if $bestAcc > currAcc$ then
16:        $\mathcal{L} \leftarrow \mathcal{L} - bestLA$;
17:        $F \leftarrow F + bestLA$;
18:        $currAcc \leftarrow bestAcc$;
19:    else
20:        break;
21:    end if
22: end while
```

in a filter set for each learning algorithm and an aggregate count (overall) is given in Table 13. MLPs and random forests are included in more than 50% of the constructed filter sets while RIPPER and NB are included in less than 2% of the filter sets. The remaining learning algorithms are used in a filter set between 13% and 23% of the time. It is interesting that some of the learning algorithms include a particular learning algorithm in the filter set for most of the data sets while other learning algorithms never or rarely include it. For example, MLP is always included in the filter set for NB, yet never for 5-NN. Also, only the MLP and 5-NN learning algorithms frequently include themselves in the filter set. Thus, hardness for a learning algorithm is often better detected using a different learning algorithm.

7 Data Set-level Analysis

Our work has focused on hardness at the instance-level. However, prior work has been done that examines what causes hardness at the data set level. The hardness measures and hardness measures can be averaged together to measure hardness at the data set level. The averaged hardness measures can provide insight into a data set's characteristics and possibly provide direction into which methods are the most appropriate for the data set. Previous studies have primarily looked at only binary classification problems. We compare instance hardness at the data set level with other data set complexity measures. We use a set of complexity measures by Ho and Basu (2002) (implemented with DCoL (Orriols-Puig et al, 2009)). In this study we do not limit our examination to two-class problems. Hence, we do not use the measurements from Ho and Basu that are only for two-class problems. Ho and Basu's complexity measures that were used are shown in Table 14. Some of the original measures were adapted to handle multi-class problems (Orriols-Puig et al, 2009).

We first compare our measures to those used by Ho and Basu (2002). The matrix of Spearman correlation coefficients comparing the hardness measures against those measures

Table 12 The average accuracy values for nine learning algorithms comparing the adaptive filtering approach against IH 0.7. "count" gives the number of times that a filtering algorithm improves, maintains, or reduces classification accuracy. The adaptive filtering approach significantly increases classification accuracy.

Algorithm	IH_0.7	A_0.5	A_0.7	A_0.9
MLP	83.583	86.302	86.863	87.997
counts		51-1-0	52-0-0	52-0-0
p-value		$\mathbf{< 0.001}$	$\mathbf{< 0.001}$	$\mathbf{< 0.001}$
C4.5	81.459	84.854	86.023	86.875
counts		49-1-2	51-1-0	52-0-0
p-value		$\mathbf{< 0.001}$	$\mathbf{< 0.001}$	$\mathbf{< 0.001}$
5-NN	82.135	85.953	87.189	89.162
counts		49-3-0	51-1-0	52-0-0
p-value		$\mathbf{< 0.001}$	$\mathbf{< 0.001}$	$\mathbf{< 0.001}$
LWL	69.649	74.382	74.043	74.048
counts		46-4-2	43-7-2	43-7-2
p-value		$\mathbf{< 0.001}$	$\mathbf{< 0.001}$	$\mathbf{< 0.001}$
NB	77.220	80.368	80.637	80.345
counts		49-2-1	50-1-1	50-1-1
p-value		$\mathbf{< 0.001}$	$\mathbf{< 0.001}$	$\mathbf{< 0.001}$
NNge	82.158	85.876	87.145	88.892
counts		49-2-1	50-1-1	50-1-1
p-value		$\mathbf{< 0.001}$	$\mathbf{< 0.001}$	$\mathbf{< 0.001}$
RandForest	83.065	86.306	87.353	89.506
counts		49-2-1	51-0-1	51-0-1
p-value		$\mathbf{< 0.001}$	$\mathbf{< 0.001}$	$\mathbf{< 0.001}$
Ridor	77.699	81.802	82.509	83.494
counts		50-1-1	51-0-1	51-0-1
p-value		$\mathbf{< 0.001}$	$\mathbf{< 0.001}$	$\mathbf{< 0.001}$
RIPPER	79.163	84.418	85.197	86.197
counts		49-1-2	50-1-1	51-0-1
p-value		$\mathbf{< 0.001}$	$\mathbf{< 0.001}$	$\mathbf{< 0.001}$
Average	79.742	83.535	84.280	85.342
counts		51-0-1	51-0-1	51-0-1
p-value		$\mathbf{< 0.001}$	$\mathbf{< 0.001}$	$\mathbf{< 0.001}$

Table 13 The frequency of selecting a learning algorithm when adaptively constructing a filter set. Each row gives the percentage of cases that the learning algorithm was included in the filter set for the learning algorithm in the column.

	Overall	MLP	C4.5	5-NN	LWL	NB	Nnge	RandF	Ridor	RIP
MLP	51.59	86.67	16.67	0	53.33	100	13.33	26.67	80.00	93.33
C4.5	17.46	13.33	16.67	6.67	26.67	0	13.33	20.00	26.67	33.33
5-NN	23.81	6.67	0	86.67	6.67	0	26.67	26.67	20.00	26.67
LWL	15.87	0	0	40.00	0	66.67	0	6.67	20.00	0
NB	1.59	0	0	0	13.33	0	0	0	0	0
NNge	18.25	26.67	16.67	20.00	13.33	0	46.67	93.33	13.33	0
RandF	55.56	26.67	100	80.00	6.67	0	80.00	0	86.67	53.33
Ridor	13.49	13.33	50.00	0	6.67	53.33	6.67	6.67	6.67	0
RIP	0.79	0	0	0	0	6.67	0	0	0	0

used by Ho and Basu are shown in Table 15. The measures were normalized by subtracting the mean and dividing by the standard deviation. The values in bold represent correlations with a magnitude greater than 0.75. Only N1 and N3 are strongly correlated with *k*DN, CL, and CLD. N1 is the percentage of instances with at least one nearest neighbor of a different class. N3 is the leave-one-out error of the one-nearest neighbor classifier. Both N1 and N3

Table 14 List of complexity measures from Ho and Basu (2002).

F2:	**Volume of overlap region**:The overlap of the per-class bounding boxes calculated for each attribute by normalizing the difference of the maximum and minimum values from each class.
F3:	**Max individual feature efficiency**: For all of the features, the maximum ratio of the number of instances not in the overlapping region to the total number of instances.
F4:	**Collective feature efficiency**: F3 only return the ratio for the attribute that maximizes the ratio. F4 is a measure for all of the attributes.
N1:	**Fraction of points on class boundary**: The fraction of instances in a data set that are connected to their nearest neighbors that have a different class in a spanning tree.
N2:	**Ratio of ave intra/inter class NN dist**: The average distance to the nearest intra-class neighbors divided by the average distance to the nearest inter-class neighbors.
N3:	**Error rate of 1NN classifier**: Leave-one-out error estimate of 1NN.
T1:	**Fraction of maximum covering spheres**: The normalized count of the number of clusters of instances containing a single class
T2:	**Ave number of points per dimension**: Compares the number of instances to the number of features.

Table 15 Spearman correlation matrix comparing the hardness measures against the complexity measures from Ho and Basu. The strong correlation (bolded values) indicate that there is some overlap between our measures and those by Ho and Basu.

	F2	F3	F4	N1	N2	N3	T1	T2
DN	0.433	0.112	0.237	**0.908**	0.696	**0.867**	0.298	-0.142
DS	-0.550	0.063	0.011	-0.523	-0.427	-0.464	-0.123	-0.445
DCP	-0.542	0.086	0.107	-0.661	-0.456	-0.676	-0.147	-0.033
TD_P	0.403	0.014	0.079	0.283	0.235	0.233	-0.040	0.340
TD_U	0.306	0.121	0.233	0.336	0.221	0.240	-0.071	0.338
CL	-0.490	0.008	-0.186	**-0.797**	-0.644	**-0.763**	-0.246	-0.078
CLD	-0.463	-0.035	-0.162	**-0.805**	-0.649	**-0.775**	-0.272	-0.063
MV	0.424	0.336	0.162	0.307	0.304	0.318	0.163	-0.148
CB	0.148	0.201	0.008	-0.034	0.062	0.015	0.065	-0.027

Table 16 The Spearman correlation coefficients for each hardness measure and Ho and Basu's complexity measures relating to data set hardness. The measures that measure class overlap have a strong correlation with data set hardness.

*k*DN	DS	DCP	TD_P	TD_U	CL	CLD	MV	CB	Lin
0.901	-0.561	-0.758	0.427	0.354	**-0.864**	**-0.868**	0.313	0.088	**0.882**

F2	F3	F4	N1	N2	N3	T1	T2	Lin
0.455	0.078	0.190	**0.860**	0.675	**0.828**	0.222	-0.127	**0.844**

are similar and can be categorized as measuring class separability. N1, N3, *k*DN, CL, and CLD measure class overlap using all of the features in the data set.

We examined each hardness measure and complexity measure individually to determine how well it predicts data set hardness (the average instance hardness of the instances in the data set). The Spearman correlation coefficient for the hardness measures and the measures from Ho and Basu with data set hardness are shown in Table 16. *k*DN, CL, CLD, N1, and N3 all have a correlation coefficient greater than 0.8. Recall that *k*DN, CL, CLD are strongly correlated with N1 and N3. Despite diversity in the measures, only these few are strongly correlated with data set hardness and they measure class overlap.

We also apply linear regression to evaluate data set hardness as a combination of the hardness measures and the measures from Ho and Basu. The correlation coefficients are shown in the column "Lin" in Table 16. For the linear model of the hardness measures, only *k*DN is statistically significant for the hardness measures. For Ho and Basu's complexity measures, only N1 is statistically significant. The correlation of data set hardness with the linear models of the hardness measures and the measures from Ho and Basu are weaker than the correlation of data set hardness with an individual measure. When using both sets of measures, the resulting correlation coefficient is 0.896 with none of the measures being statistically significant. The linear model also has a weaker correlation coefficient than only using *k*DN.

Based on correlation from a linear regression model, our aggregate hardness measures are competitive with those from Ho and Basu. When the hardness measures are used in combination with those from Ho and Basu, a slightly stronger correlation is achieved. This is somewhat expected as there are many underlying and misunderstood factors that affect complexity. By measuring the complexity from many different angles, more perspective can be found.

The averaged hardness measures at the data set level provide an indication of the source of hardness and could further indicate which learning algorithms and/or methods for integrating instance hardness into the learning process are the most appropriate to use for a particular data set. A cursory examination of the correlation of the hardness measures with instance hardness at the data set level (Table 16) does not reveal an obvious connection. Further in depth analysis is left for future work.

8 Related Work

There are a number of methods and approaches that can be used to identify instances that are hard to classify correctly. In this section we review some previous work for identifying hard instances. Fundamentally, instances that are hard to classify correctly are those for which a learning algorithm has a low probability of predicting the correct class label after having been trained on a training set. To compare the related works with instance hardness we reference the hypothetical data set in Figure 2. For convenience, we reproduce Figure 2 in Figure 4. We also compare instance hardness (IH_ind and IH_class) with related works in Table 17 on a subset of the examined instances. The columns under "IH_h–Classifier Scores" are instance hardness values calculated using the classifier score for a specific learning algorithm. Table 17 is divided into three sections: the first section contains instances with high instance hardness (IH_ind ~ 1), the second section contains instances with low instance hardness (IH_ind ~ 0), and the third section contains instances with instance hardness around 0.5. We will refer to Table 17 throughout this section.

Machine learning research has observed that data sets are often noisy and contain outliers, and that noise and outliers are harder to classify correctly. Although we do not explicitly search for outliers, outliers and noisy instances will constitute a subset of the hard instances. Much work has been put forth to identify outliers and noise. Discovering outliers is important in anomaly detection where an outlier may represent an important instance. For example, an outlier in a database of credit card transactions may represent a fraudulent transaction. Anomaly detection is generally unsupervised and looks at the data set as a whole. One of the difficulties with outlier detection is that there is no agreed-upon definition of what constitutes an outlier or noise. Thus, a variety of different outlier detection methods exist, such as statistical methods (Barnett and Lewis, 1978), distance-based methods (Knorr

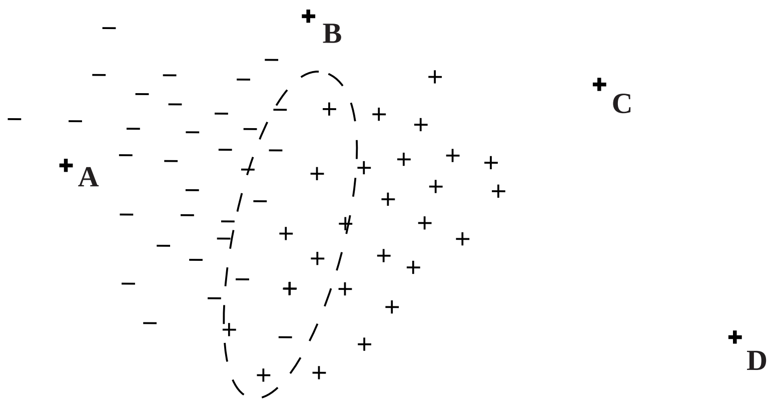

Fig. 4 Hypothetical 2-dimensional data set.

Table 17 Comparison summary of the methods that identify hard instances. The first section of the table shows examples of instances that have high instance hardness values (IH = 1), the second section shows examples that are easy but the LoOP value is high (IH$\sim$ 0 && LOP$\sim$ 1), the third section has instances with medium instance hardness (IH$\sim$0.5). The active learning values are the uncertainty scores–higher values represent more uncertainty; the outlier detection values are "yes" if the instance is an outlier and "no" if it is not–for LOP, higher values indicate a higher likelihood of being an outlier. The values in bold represent values that are unexpected given the instance hardness value. For example, an instance with an instance hardness value of 1 is expected to be considered an outlier. On the other hand, it is unexpected when an instance with a low instance hardness value ($\sim$ 0) is categorized as an outlier instance.

data		IH		IH_h–Classifier Scores									Active Learn			Outlier Detection			
set	id	ind	class	MLP	C4.5	IB5	LWL	NB	NNg	Rand	Rid	RIP	USN	QN	QC	LOP	Maj	Con	REN
ecoli	263	1	0.85	1	0.78	1	0.91	1	1	0.92	1	0.96	0.99	0	0.72	**0.26**	yes	**no**	yes
ta	128	1	0.82	0.76	0.90	0.99	0.78	0.95	1	0.98	1	0.85	0.63	0.14	**0.52**	**0**	yes	**no**	yes
abal	4014	1	0.86	0.94	1	1	0.91	1	1	1	1	0.87	0.04	**0.46**	**0.47**	**0.04**	yes	yes	**no**
yeast	327	1	0.86	0.99	0.99	1	0.87	0.95	1	0.88	1	0.88	**0.63**	0.17	**0.42**	**0.22**	yes	yes	**no**
abal	720	1	0.89	1	1	1	1	1	1	1	1	1	0.99	0.08	**0.32**	1	yes	yes	**no**
spam	3913	0.07	0.09	**0.59**	0	0	0.20	0	0	0.02	0	0.05	**1**	0	0.15	**1**	no	no	no
splice	1807	0	0.03	0	0	0	0.11	0	0	**0.39**	0	0.01	**0.99**	0	0	**0.96**	no	no	**yes**
adult	30676	0.04	0.05	0.05	0.02	0	0.06	0.00	**0.40**	0.11	0	0.13	**1**	0	0	**1**	no	no	no
adult	30833	0	0.00	0.11	0.08	0	0.07	0.03	0	0	0	0.13	**0.94**	0	0	**1**	no	no	no
wave	3524	0.58	0.53	0.66	**0.98**	**1**	**0.81**	**0.19**	0	0.58	0.60	0.64	0.60	0	0.54	**0.34**	yes	yes	yes
chess	8691	0.53	0.53	**0.82**	0.72	0.58	**0.87**	**0.81**	0	0.46	0.40	0.48	0.04	0.57	0.38	**0**	yes	no	no
annea	715	0.53	0.56	0.02	0.64	**1**	0.56	0.67	**0.80**	0.76	0.40	**0.16**	0.18	0.68	0.73	**0.01**	yes	yes	yes
arrhy	69	0.51	0.57	**0.92**	0.45	**1**	0.43	**0**	**1**	0.76	0	0.22	1	**0.95**	0.42	**0.18**	yes	yes	yes

and Ng, 1999), and density-based methods (Breunig et al, 2000). Anomaly detection methods identify anomalous instances as those that lie outside the group(s) of the majority of the other instances in the data set. In the hypothetical two-dimensional data set shown in Figure 4, instances C and D would be identified as anomalous but not instances A and B.

Most anomaly detection methods do not have a continuous output and are not supervised. One anomaly detection method that outputs continuous values is local outlier factor. Local outlier factor (LOF) (Breunig et al, 2000) suggests that each instance has a degree of "outlierness" rather than a binary labeling. LOF seeks to overcome the problem facing most anomaly detection methods–that the sub-spaces within many data sets have different densities. LOF considers relative density rather than the global density of the data set. Instances with a LOF value of 1 or less are not outliers. The values produced by LOF are somewhat hard to interpret as there is no upper bound or any value that indicates when a LOF value represents an outlier. For one data set, an LOF value of 1.1 may represent an outlier while a value of 2 could represent an outlier in another data set.

There are a number of approaches that aim at overcoming the uninterpretability of LOF (Kriegel et al, 2011). One approach is local outlier probability (LoOP) (Kriegel et al, 2009). LoOP builds on LOF with statistical methods to produce a probability that an instance is a local density outlier. This allows the values to be compared across data sets. There are two major assumptions that LoOP makes: 1) that the k-nearest neighbors of an instance p are centered around p and 2) that the distances behave like the positive leg of a normal distribution. Despite being more interpretable, LoOP often does not identify hard instances as outliers and identifies easy instances as outliers as shown in Table 17 (LOP). Low values indicate that an instance is not likely to be an outlier according to LoOP.

Filtering, or removing instances prior to training, is another approach that seeks to identify mislabeled and/or noisy instances with the intent of improving an inferred model of the data. Unlike anomaly detection, filtering is often supervised, removing instances that are misclassified by a learning algorithm. In Figure 4, filtering would likely identify instances A, B, and some of the border points as hard to classify. A popular approach to outlier detection is repeated-edited nearest-neighbor (RENN) (Tomek, 1976) which repeatedly removes the instances that are misclassified by a 3-nearest neighbor classifier and has produced good results. Brodley and Friedl (1999) expanded this idea by removing the instances that were misclassified by all or the majority of the learning algorithms in an ensemble of three learning algorithms. These methods do not output a continuous value but they do take into account the class label. As shown in Table 17, these methods (maj, con, and REN) do not often identify easy instances as outliers, but they may not identify hard instances as outliers.

Some learning algorithms produce probabilistic output, such as naïve Bayes, Bayes nets, and Gaussian processes. The output from probabilistic algorithms could naturally answer the question of which instances are hard to classify correctly. However, there are often assumptions that are made that are not true of the data distribution (i.e. the attributes are independent or the data is normally distributed). Many other machine learning algorithms do not produce a probabilistic output. In those cases, the probabilities can be approximated by normalizing the output values or using some heuristic to produce pseudo probabilistic values. The posterior classifier probabilities from the learning algorithms in $\mathcal{L}$ for a subset of the instances are provided in Table 17. The posterior classifier probabilities provide a good approximation for instance hardness, but, as discovered in Section 5, they have a lower correlation with the hardness measures. This is apparent when examining instances that have an instance hardness measure around 0.5 (the last four instances in Table 17).

Probabilistic outputs from a classifier are important when the outputs are combined with other sources of information for making decisions, such as the outputs from other classifiers. Probabilistic outputs are often not well calibrated, such as the output from naïve Bayes (Domingos and Pazzani, 1996). As such, a number of methods have been proposed to calibrate classifier scores (Bennett, 2000; Platt, 2000; Zadrozny and Elkan, 2001, 2002). For binary classification problems, the calibration is usually done by training the learning algorithm to get the classifier scores $s(x)$ and then mapping these scores into a probability estimate $\hat{P}(y|x)$ by learning a mapping function. Platt (2000) suggests finding the parameters A and B for a sigmoid function of the form $\hat{P}(y|x) = \frac{1}{1+e^{As(x)+B}}$ to map the classifier scores $s(x)$ to the probability estimates minimizing the log-likelihood of the data. Multi-class classification problems are broken down into binary classification problems such as 1 vs 1 or 1 vs all. 1 vs 1 creates a classifier for each pair of classes. 1 vs all creates a classifier that discriminates between the instances of a particular class and all the instances that have a different class. The calibrated probabilities from the binary classification problems are then recombined back together. Classifier scores are supervised and produce continuous outputs for identifying hard instances. In Figure 4, instances A, B, and the border points would be identified as being hard to classify.

Active learning (Settles, 2010) seeks to find the most informative instances in a data set. Active learning assumes that there is a set of labeled instances and an abundance of unlabeled training data and that labeling the data is expensive, thus, the most informative instances should be labeled first. In active learning, a learning algorithm chooses which instances to use for training. Active learning assigns unlabeled instances a degree of how informative they may be to a learning algorithm by optimizing a given criterion. This informative measure could be used as a means of identifying hard instances. For example, uncertainty sampling (Lewis and Gale, 1994) selects an unlabeled instance x^* whose labeling the learning algorithm is least certain about:

$$x^* = \underset{x}{\operatorname{argmax}}\, 1 - p(\hat{y}|x)$$

where $\hat{y}$ is the class label with the highest posterior probability for the learning algorithm. Other methods, such as query-by-committee (Seung et al, 1992; Freund et al, 1992) and a Support Vector Machine method by Tong and Koller (2001), seek to reduce the size of the version space[6] (Mitchell, 1982). Query-by-committee uses a committee of models trained on the labeled instances and selects the instances that the committee disagrees about most. Thus, active learning identifies the border points as being hard to classify. Table 17 shows that active learning scores vary widely for the same instances. For instances with an instance hardness value near 1, it would be expected that the uncertainty value would be close to either 1 or 0. Since the class is not included for active learning, a hard instance would appear to be the wrong class (i.e. the instance is mislabeled) or it would have high uncertainty. For the easy instances, a low uncertainty value would be expected. Active learning scores do not have a high correlation with instance hardness.

Clearly, none of the previous work was designed to better understand why instances are misclassified as is the case with instance hardness. For example, filtering aims at removing mislabeled instances from a data set, and the classifier scores are for applications where a confidence on a prediction is required. Incorporating the ideas of previous work, instance hardness provides a framework for identifying which instances are hard to classify and understanding why they are hard to classify.

9 Conclusions and Future Work

In this paper we examined why instances are misclassified and how to characterize them. We presented instance hardness to empirically discover which instances are hard to classify correctly. We also presented a set of hardness measures to characterize why some instances are difficult to classify correctly. We used a broad set of data sets and learning algorithms and examined hardness at the instance level. We found that class overlap is a principal contributor to instance hardness and data set hardness. The hardness measures *k*DN, CL, and CLD capture class overlap and are strongly correlated with instance hardness. Class skew has been observed to increase instance hardness. We found that class skew alone does not

[6] Version space is the subset of parameters that correctly classifies the labeled examples.

cause instances to be misclassified. Rather, class skew exacerbates the other characteristics, such as class overlap, that cause an instance to be misclassified as demonstrated when the instances and their hardness values were segregated according to their MV value (Table 8). Continued study of instance hardness should lead to additional insights regarding data complexity.

Being able to measure instance hardness and complexity has important ramifications for future machine learning and meta-learning research. We briefly examined integrating instance hardness into the learning process by filtering the data sets prior to training and using informative error. In each case, integrating into the learning process the knowledge of which instances are hard to classify correctly resulted in a significant increase in classification accuracy. As a specific example, informative error significantly increased the classification accuracy over various filtering and boosting approaches. These techniques show that integrating into the learning process the knowledge about which instances are hard can increase generalization accuracy. Future work includes understanding the circumstances and situations which are most appropriate for each technique. There is no one technique for identifying hard instances that is best for all data sets as demonstrated with the adaptive filter sets.

Calculating instance hardness and the hardness measures can be a computationally expensive procedure. Despite requiring the computation of N learning algorithms, the instance hardness values only need to be computed once and they can be used in a wide variety of applications as was shown in Section 6. The hardness measures need to be calculated only once as well. For many data sets, this additional computational complexity is acceptable. For massive data sets, though, the additional computational complexity can be a significant concern. In this case, the set of learning algorithms used to calculate instance hardness and the hardness measures can be altered to those that better handle massive data sets. Also, we showed that there is no specific set of learning algorithms that is best for all data sets and learning algorithms. Using the same learning algorithm to calculate instance hardness and to infer the model of the data does not always result in the most accurate model.

Being able to better analyze data would allow a practitioner to select an algorithm more suited to their purposes. Also, the evaluation of a learning algorithm could be enhanced by knowing which instances are hard and, with a high likelihood, will be misclassified. This could lead to a better stopping criterion. We expect that the exploration of instance hardness and data complexity may lead to more in depth investigation and applications in new areas of machine learning and data mining. Instance hardness and the hardness measures could be used in combination with techniques from active learning to determine a subset of the most important instances from a data set. Future work also includes work in meta-learning. For example, the hardness measures could be used to estimate the performance of a learning algorithm on a data set.

References

Abe N, Mamitsuka H (1998) Query learning strategies using boosting and bagging. In: Proceedings of the Fifteenth International Conference on Machine Learning, pp 1–9

Abe N, Zadrozny B, Langford J (2006) Outlier detection by active learning. In: Proceedings of the 12th international conference on Knowledge discovery and data mining, ACM, New York, NY, USA, pp 504–509

Barnett V, Lewis T (1978) Outliers in statistical data., 2nd edn. John Wiley & Sons Ltd.

Batista GEAPA, Prati RC, Monard MC (2004) A study of the behavior of several methods for balancing machine learning training data. SIGKDD Explorations Newsletter 6(1):20–29

Bennett PN (2000) Assessing the calibration of naive bayes posterior estimates. Tech. Rep. CMU-CS-00-155, Carnegie Mellon University

Brazdil P, Giraud-Carrier C, Soares C, Vilalta R (2009) Metalearning: Applications to Data Mining. Springer

Breunig MM, Kriegel HP, Ng RT, Sander J (2000) Lof: identifying density-based local outliers. SIGMOD Record 29(2):93–104

Bridle JS (1989) Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In: Neuro-computing: Algorithms, Architectures and Applications, Springer, pp 227–236

Brighton H, Mellish C (2002) Advances in instance selection for instance-based learning algorithms. Data Mining and Knowledge Discovery 6(2):153–172

Brodley CE, Friedl MA (1999) Identifying mislabeled training data. Journal of Artificial Intelligence Research 11:131–167

Brodley CE, Utgoff PE (1995) Multivariate decision trees. Machine Learning 19(1):45–77

Dagan I, Engelson SP (1995) Committee-based sampling for training probabilistic classifiers. In: Proceedings of the 12th International Conference on Machine Learning, pp 150–157

Domingos P, Pazzani MJ (1996) Beyond independence: Conditions for the optimality of the simple bayesian classifier. In: Saitta L (ed) ICML, Morgan Kaufmann, pp 105–112

Frank A, Asuncion A (2010) UCI machine learning repository. URL http://archive.ics.uci.edu/ml

Freund Y, Schapire RE (1996) Experiments with a new boosting algorithm. In: Thirteenth International Conference on Machine Learning, pp 148–156

Freund Y, Seung HS, Shamir E, Tishby N (1992) Information, prediction, and query by committee. In: Advances in Neural Information Processing Systems (NIPS), pp 483–490

Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH (2009) The weka data mining software: an update. SIGKDD Explorations Newsletter 11(1):10–18

Ho TK, Basu M (2002) Complexity measures of supervised classification problems. IEEE Trans Pattern Anal Mach Intell 24:289–300

van Hulse J, Khoshgoftaar TM, Napolitano A (2007) Experimental perspectives on learning from imbalanced data. In: Proceedings of the 24th international conference on Machine learning, ACM, pp 935–942

John GH (1995) Robust decision trees: Removing outliers from databases. In: Knowledge Discovery and Data Mining, pp 174–179

Knorr EM, Ng RT (1999) Finding intensional knowledge of distance-based outliers. In: Proceedings of the 25th International Conference on Very Large Data Bases, pp 211–222

Kriegel HP, Kröger P, Schubert E, Zimek A (2009) Loop: local outlier probabilities. In: Proceedings of the 18th ACM Conference on Information and Knowledge Management, pp 1649–1652

Kriegel HP, Kröger P, Schubert E, Zimek A (2011) Interpreting and unifying outlier scores. In: SDM, pp 13–24

Lee J, Giraud-Carrier C (2011) A metric for unsupervised metalearning. Intelligent Data Analysis 15(6):827–841

Lewis DD, Gale WA (1994) A sequential algorithm for training text classifiers. In: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval, pp 3–12

Mansilla EB, Ho TK (2004) On classifier domains of competence. In: ICPR (1), pp 136–139
Mitchell TM (1982) Generalization as search. Artifical Intelligence 18(2):203–226
Orriols-Puig A, Macià N, Bernadó-Mansilla E, Ho TK (2009) Documentation for the data complexity library in c++. Tech. Rep. 2009001, La Salle - Universitat Ramon Llull
Peterson AH, Martinez TR (2005) Estimating the potential for combining learning models. In: Proceedings of the ICML Workshop on Meta-Learning, pp 68–75
Platt J (2000) Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. In: Advances in Large Margin Classifiers
Quinlan JR (1993) C4.5: Programs for Machine Learning. Morgan Kaufmann, San Mateo, CA, USA
Salojärvi J, Puolamäki K, Simola J, Kovanen L, Kojo I, Kaski S (2005) Inferring relevance from eye movements: Feature extraction. Tech. Rep. A82, Helsinki University of Technology
Sayyad Shirabad J, Menzies T (2005) The PROMISE Repository of Software Engineering Databases. School of Information Technology and Engineering, University of Ottawa, Canada, URL http://promise.site.uottawa.ca/SERepository/
Scheffer T, Decomain C, Wrobel S (2001) Active hidden markov models for information extraction. In: Proceedings of the 4th International Conference on Advances in Intelligent Data Analysis, Springer-Verlag, London, UK, UK, IDA ’01, pp 309–318
Segata N, Blanzieri E, Cunningham P (2009) A scalable noise reduction technique for large case-based systems. In: Proceedings of the 8th International Conference on Case-Based Reasoning: Case-Based Reasoning Research and Development, pp 328–342
Settles B (2010) Active learning literature survey. Tech. Rep. Computer Sciences Technical Report 1648, Univeristy of Wisconsin-Madison
Seung HS, Opper M, Sompolinsky H (1992) Query by committee. In: Proceedings of the fifth annual workshop on Computational learning theory, pp 287–294
Smith MR, Martinez T (2011) Improving classification accuracy by identifying and removing instances that should be misclassified. In: Proceedings of the IEEE Internation Joint Conference on Neural Networks, pp 2690–2697
Stiglic G, Kokol P (2009) GEMLer: Gene expression machine learning repository. University of Maribor, Faculty of Health Sciences, URL http://gemler.fzv.uni-mb.si/
Thomson K, McQueen RJ (1996) Machine learning applied to fourteen agricultural datasets. Tech. Rep. 96/18, The University of Waikato
Tomek I (1976) An experiment with the edited nearest-neighbor rule. IEEE Transactions on Systems, Man, and Cybernetics 6:448–452
Tong S, Koller D (2001) Support vector machine active learning with applications to text classification. Journal of Machine Learning Research 2:45–66
Webb GI (2000) Multiboosting: A technique for combining boosting and wagging. Machine Learning 40(2):159–196
Wolpert DH (1996) The lack of a priori distinctions between learning algorithms. Neural Computation 8(7):1341–1390
Zadrozny B, Elkan C (2001) Learning and making decisions when costs and probabilities are both unknown. In: KDD, pp 204–213
Zadrozny B, Elkan C (2002) Transforming classifier scores into accurate multiclass probability estimates. In: KDD, ACM, pp 694–699