Identifying and Eliminating Mislabeled Training Instances

Carla E. Brodley
School of Electrical
and Computer Engineering
Purdue University
West Lafayette, IN 47906
brodley@ecn.purduc.edu

Mark A. Friedl
Department of Geography and
Center for Remote Sensing
Boston University
Boston, MA 02215
friedl@crsa.bu.edu

Abstract

This paper presents a new approach to identifying and eliminating mislabeled training instances. The goal of this technique is to improve classification accuracies produced by learning algorithms by improving the quality of the training data. The approach employs an ensemble of classifiers that serve as a *filter* for the training data. Using an n-fold cross validation, the training data is passed through the filter. Only instances that the filter classifies correctly are passed to the final learning algorithm. We present an empirical evaluation of the approach for the task of automated land cover mapping from remotely sensed data. Labeling error arises in these data from a multitude of sources including lack of consistency in the vegetation classification used, variable measurement techniques, and variation in the spatial sampling resolution. Our evaluation shows that for noise levels of less than 40%, filtering results in higher predictive accuracy than not filtering, and for levels of class noise less than or equal to 20% filtering allows the base-line accuracy to be retained. Our empirical results suggest that the ensemble filter approach is an effective method for identifying labeling errors, and further, that the approach will significantly benefit ongoing research to develop accurate and robust remote sensing-based methods to map land cover at global scales.

Introduction

A goal of an inductive learning algorithm is to form a generalization from a set of training instances such that classification accuracy on previously unobserved instances is maximized. The maximum accuracy achievable depends on the quality of the data and on the appropriateness of the biases of the chosen learning algorithm for the data. The work described here focuses on improving the quality of the training data by identifying and eliminating mislabeled instances prior to applying the chosen learning algorithm, thereby increasing classification accuracy.

For some learning tasks, domain knowledge exists such that noisy instances can be identified because they go against the "laws" of the domain. For example, in the domain of diagnosing Alzheimer's disease, it is known that the illness strikes the elderly. An instance, describing a patient, labeled as sick (versus not sick) for which the patient's age is ten is clearly incorrect. This is an example of an instance for which the class label is incorrect, or a faulty measurement of the age feature was recorded. For many domains, this type of knowledge does not exist, and an automated method is needed to eliminate mislabeled instances from the training data.

The idea of eliminating instances to improve the performance of nearest neighbor classifiers has been a focus of research in both pattern recognition and instance-based learning. Wilson (1972) used a three-nearest neighbor classifier (3-NN) to select instances that were then used to form a 1-NN; only instances that the 3-NN classified correctly were retained for the 1-NN. Aha, Kibler and Albert (1991) demonstrated that filtering instances based on records of their contribution to classification accuracy in an instance-based classifier improves the accuracy of the the resulting classifier. Skalak (1994) created an instance selection mechanism for nearest neighbor classifiers with the goal of reducing their computational cost, which depends on the number of stored instances.

The idea of selecting "good" instances has also been applied to other types of classifiers. Winston (1975) demonstrated the utility of selecting "near misses" when learning structural descriptions. Skalak & Rissland (1990) describe an approach to selecting instances for a decision tree algorithm using a case-based retrieval algorithm's taxonomy of cases (for example "the most-on-point cases"). Lewis and Catlett (1994) illustrate that sampling instances using an estimate of classification certainty drastically reduces the amount of data needed to learn a concept.

In this article we address the problem of identify-

ing training instances that are mislabeled. Quinlan (1986) demonstrated that, for higher levels of noise, removing noise from attribute information decreases the predictive accuracy of the resulting classifier if the same attribute noise is present when the classifier is subsequently used. In the case of mislabeled training instances (class noise) the opposite is true; *cleaning* the training data will result in a classifier with higher predictive accuracy. Cleaning can take one of two forms: removing mislabeled instances from the training data, or correcting their labels and retaining them.

In the next section we introduce a method for identifying mislabeled instances that is not specific to any single learning algorithm, but rather serves as a general method that can be applied to a dataset before feeding it to a specific learning algorithm. The basic idea is to use a set of learning algorithms to create classifiers that serve as a *filter* for the training data. The method was motivated by the technique of removing *outliers* in regression analysis (Weisberg, 1985). An outlier is a case (an instance) that does not follow the same model as the rest of the data, appearing as though is comes from a different probability distribution. Candidates are cases with a large residual error.[1] Weisberg suggests building a model using all of the data except for the suspected outlier and testing whether it does or does not belong to the model using the externally studentized t-test.

Here, we apply this idea by using a set of classifiers formed from part of the training data to test whether instances in the remaining part of the training data are mislabeled. An important difference between our work and previous approaches to outlier detection is that our approach assumes that the errors in the class labels are independent of the particular model being fit to the data. In essence, our method attempts to identify data points that would be outliers in *any* model.

Filtering Training Data

This section describes a general procedure for identifying and eliminating mislabeled instances from a training set. The first step is to identify candidate instances by using m learning algorithms (called *filter algorithms*) to tag instances as correctly or incorrectly labeled. To this end, a n-fold cross-validation is performed over the training data. For each of the n parts, the m algorithms are trained on the other $n-1$ parts. The m resulting classifiers are then used to tag each instance in the excluded part as either correct or mislabeled. An individual classifier tags an instance as mislabeled if it cannot classify the instance correctly.

At the end of the n-fold cross-validation each instance in the training data has been tagged. Using this information, the second step is to form a classifier using a new version of the training data for which all of the instances identified as mislabeled are removed. The filtered set of training instances is provided as input to the *final learning algorithm*. The resulting classifier is the end product of the approach. Specific implementations of this general procedure differ in how the filtering is performed, and in the relationship between the filter algorithm(s) and the final algorithm.

Implementing the General Procedure

One approach to implementing this procedure is to use the same algorithm to construct both the filter and the final classifier. This approach is most similar to removing outliers in regression analysis, for which the same model is used to test for outliers and for fitting the final model to the data once the outliers have been removed. A related approach method is John's (1995) method for removing the training instances that are pruned by C4.5 (Quinlan, 1993). After the instances are removed a new tree is constructed using the filtered training data. In this approach training instances are filtered based on C4.5's pruning decisions, whereas our approach filters instances based on classification decisions.

A second way to implement filtering is to construct the filter using one algorithm and the final classifier using a different algorithm. The assumption underlying this approach is that some algorithms act as good filters for other algorithms, much like some algorithms act as good feature selection methods for others (Cardie, 1993). Wilson's (1972) approach to filtering data for a 1-NN using a 3-NN is an example of this approach.

A third method is based on ensemble classifiers, which combine the outputs of a set of base-level classifiers (Hansen & Salamon, 1990; Benediktsson & Swain, 1992; Wolpert, 1992). A majority vote ensemble classifier will outperform each individual base-level classifier on a dataset if two conditions hold: (1) the probability of a correct classification by each individual classifier is greater than 0.5 and (2) if the errors in predictions of the base-level classifiers are independent (Hansen & Salamon, 1990). For this work, we use an ensemble classifier to detect mislabeled instances by constructing a set of *base-level detectors* (classifiers) and then using them to identify mislabeled instances by consensus vote. This is distinct from majority vote in that *all* base-level detectors must agree that an instance is mislabeled for it to be eliminated from the training

[1] Not all residual cases are outliers because according to the model, large errors will occur with the frequency prescribed by the generating probability distribution.

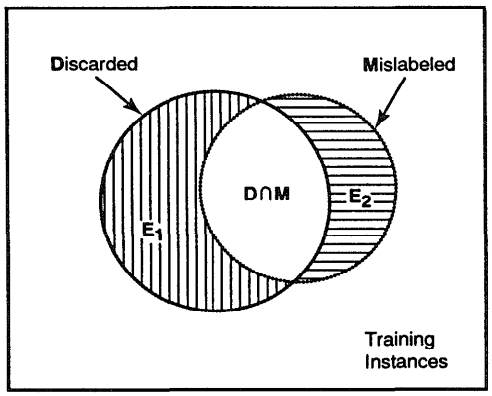

Figure 1: Types of detection errors

data.

Consensus Filters

In regression analysis, outliers are defined relative to a particular model. Here we assume that some instances in the data have been mislabeled and that the label errors are independent of the particular model being fit to the data. Therefore collecting information from different models will provide a better method for detecting mislabeled instances than from a single model.

Training data (ground truth) for the purposes of land cover mapping is generally scarce. Indeed, this problem is common in many classification and learning problem domains (e.g. medical diagnosis). Therefore, we want to minimize the probability of discarding an instance that is an exception rather than an error. Indeed, Danyluk and Provost (1993) note that learning from noisy data is difficult because it is hard to distinguish between noise and exceptions, especially if the noise is systematic. Ideally, the biases of at least one of the learning algorithms will enable it to learn the exception. Therefore, one or more of the classifiers that comprise the base-level set of detectors can have difficulty capturing a particular exception without causing the exception to be erroneously eliminated from the training data. Taking a consensus rather than a majority vote is a more conservative approach and will result in fewer instances being eliminated from the training data. The drawback of a conservative approach is the added risk in retaining bad data.

In identifying mislabeled instances there are two types of error that can be made (see Figure 1). The first type (E_1) occurs when an instance is incorrectly tagged as mislabeled (D). The second type of error (E_2) occurs when a mislabeled instance (M) is tagged as correctly labeled.

A consensus filter has a smaller probability of making an E_1 error than each of its base-level detectors if the errors made by the base-level detectors are independent. Let p_i be the probability that a base-level detector i makes an error of type E_1, then the probability that a consensus filter comprised of m base-level detectors will make an error is:

$$P(E_1) = \prod_{i=1}^{m} p_i$$

The probability of mistaking a mislabeled instance for a correctly labeled instance (E_2) is computed differently. Let q_i be the probability that a base-level detector i makes an error of type E_2. A consensus filter makes a type E_2 error if one or more of the base-level classifiers makes a type E_2 error. The probability that a consensus filter makes an E_2 error is given by:

$$q_i \leq P(E_2) \leq \sum_{i=1}^{m} q_i$$

The lower bound represents the case where each classifier makes identical errors. The upper bound represents the case where each base-level classifier makes type E_2 errors on different parts of the data. Therefore, in direct contrast to type 1 errors, independence of the errors can lead to higher overall E_2 error.

Empirical Evaluation

This research was motivated by the uncertainty caused by labeling errors in land-cover maps of the Earth's surface. In this context, we evaluate the ability of consensus filters to identify mislabeled training instances in remotely sensed data that was labeled using existing land cover maps. To simulate the type of error that is common to land-cover maps, we artificially introduced noise between pairs of classes that are likely to be confused in the original labels. We chose not to introduce noise between all pairs of classes as this would not model the types of labeling errors that occur in practice. Our experiments are designed to evaluate the consensus filter's ability to identify mislabeled instances and the effect that eliminating mislabeled instances has on predictive accuracy.

Automated Land Cover Mapping

The dataset consists of a time series of globally distributed satellite observations of the Earth's surface. The dataset was compiled by Defries and Townsend (1994), and includes 3398 locations that encompass all major terrestrial biomes[2] and land cover types at the Earth's surface (see Table 1).

[2] A *biome* is the largest subdivision of the terrestrial ecosystems. Some examples of biomes are grasslands, forests and deserts.

Table 1: Land cover classes

	Class Name	Insts
1	broadleaf evergreen forest	628
2	conif. evergreen forest & woodland	320
3	high lat. decid. forest & woodland	112
4	tundra	735
5	decid.-evergreen forest & woodland	57
6	wooded grassland	212
7	grassland	348
8	bare ground	291
9	cultivated	527
10	broadleaf decid. forest & woodland	15
11	shrubs and bare ground	153

The remote sensing observations are measurements of a parameter called the normalized difference vegetation index (NDVI). This index is commonly used to infer the amount of live vegetation present within a pixel at the time of data acquisition. Each one degree pixel is described by a time series of twelve NDVI values at monthly time increments from 1987, and by its latitude, which can be useful for discriminating among classes with otherwise similar spectral properties.

Learning Algorithms

We used three well-known algorithms from the machine learning and statistical pattern recognition communities: decision trees, nearest neighbor classifiers and linear machines. The decision tree algorithm uses the information gain ratio (Quinlan, 1986) to construct the tree, and prunes the tree using C4.5's pruning algorithm with a confidence level of 0.10. We choose to set $k = 1$ for the k-nearest neighbor algorithm, but in future implementations we will experiment with varying values of k. To find the weights of a linear machine (Nilsson, 1965) we used the thermal training rule for linear machines (Brodley & Utgoff, 1995).

Experimental Method

To test the data filtering procedure described above, we introduced random noise into the training data between pairs of classes that are most likely to be confused in the original labels. In this way, we have realistically simulated a type of labeling error that is common to land cover maps. This type of error occurs because discrete classes and boundaries are used to distinguish between classes that have transitional boundaries in space and that have fairly small differences in terms of their physical attributes. For example, the distinction between a grassland and wooded grassland can be subtle. Consequently, pixels labeled as grassland may in fact represent open woodland areas and vice versa, especially at the one degree spatial resolution of the data used here. For this work, we introduced random error between the following pairs of classes: 3-4, 5-2, 6-7, 8-11, 5-10.

For each of ten runs, the dataset was divided into a training (90%) set and a testing (10%) set. For each run, an even distribution over the classes was enforced to reduce variation in performance across different runs. After the data was split into independent train and test sets, we then corrupted the training data by introducing labeling errors. For a noise level x, an individual observation whose class is one of the identified problematic pairs has an x% chance of being corrupted. For example, an instance from class 8 (bare ground) has an x% chance of being changed to class 11 (shrubs and bare ground), and an instance from class 11 has an x% chance of being changed to class 8. Using this method the percentage of the entire training set that is corrupted will be less than x% because only some pairs of classes are considered problematic. The actual percentage of noise in the corrupted training data is reported in column 2 of Table 2.

For each of six noise levels, ranging from 0% to 40%, we compared the average predictive accuracy of classifiers trained using filtered and unfiltered data. For each of the ten runs that make up the average, we used a four-fold cross-validation to filter the corrupted instances from the training data. The consensus filter consisted of the following base-level classifiers: a decision tree, a linear discriminant function and a 1-NN classifier. To assess the ability of the consensus filter to identify the corrupted instances we then trained each of the three algorithms twice: first using the unfiltered dataset then using the filtered dataset. In addition, we formed two majority vote ensemble classifiers: one from the filtered and one from unfiltered data. The majority vote ensemble serves as the *final classifier* and not as the filter. The resulting classifiers were then used to classify the uncorrupted test data.

Affect of Filtering on Classification Accuracy

Table 2 reports the accuracy of the classifiers formed by each of the three algorithms without a filter (none) and with a consensus filter (CF). When zero noise is introduced, filtering did not make a significant difference for any of the methods. Since the original data is not guaranteed to be noise free, we have no way to evaluate whether it improves the *true* classification accuracy by using the test data. For noise levels of 5 to 30%, filtering significantly improved the classification accuracy (at the 0.05 level of significance using a paired

Table 2: Comparison of classification accuracy of filtered versus unfiltered data

Noise	Actual	1-NN		LM		UTree		Majority	
Level	Noise	None	CF	None	CF	None	CF	None	CF
0	0.0	87.3	87.5	78.6	80.0	85.6	85.5	87.3	87.1
5	3.4	84.4	87.8	76.7	78.9	84.4	86.1	86.5	87.2
10	7.1	81.8	86.4	77.5	79.2	80.1	85.1	84.4	86.7
20	13.8	75.8	83.1	70.2	78.1	75.2	81.8	80.7	85.6
30	23.3	68.6	75.2	63.4	74.0	67.4	74.2	71.7	79.0
40	36.1	58.4	59.9	49.0	54.2	56.9	59.6	57.5	59.7

t-test) in all cases except when a linear machine was the final classifier under noise levels 5% and 10%. For noise levels of 5% and 10% the filtering allows retention of approximately the same accuracy as the original uncorrupted dataset. At 20% noise the accuracies of k-NN and the decision tree constructed from the filtered data begin to drop, but not as substantially as when they are constructed from the unfiltered data. For example, applying a decision tree to the unfiltered data set causes a drop of 12.2% (100 * (85.6 - 75.2)/ 85.6) from the base-line accuracy versus a 4.4% drop when using the filtered data. At 30% noise, filtering cannot fully overcome the error in the data and for noise levels of 40% and over, consensus filtering does not help.

A hypothesis of interest is whether a majority vote ensemble classifier can be used instead of filtering. The final column of Table 2 reports the accuracies of majority vote ensembles constructed from unfiltered and filtered data. Each ensemble consists of three base-level classifiers: a 1-NN, a linear machine and a univariate decision tree. During classification, if no majority class was predicted for an instance, then the ensemble selects the class predicted by the univariate decision tree. The results show that the majority ensemble classifier formed from the unfiltered dataset is not as accurate as the 1-NN and the decision tree formed from filtered data (excluding the case of a tree constructed with 5% noise). Using a consensus filter in conjunction with a majority vote ensemble results in higher accuracy than any of the other methods in all but three cases (1-NN for noise levels 0, 5 and 40). In summary, the results show that the consensus filter improves the classification accuracies of all four learning methods.

Another point worth noting is that applying the consensus filter to the training data leads to substantially smaller decision trees. Table 3 reports the number of leaves in decision trees produced from the filtered and unfiltered data. For 0-10% noise, the filtered data creates a tree with fewer leaves than a tree induced from the original dataset. This affect was also observed by John (1995) and attributed to *Robust C4.5's* ability to remove "confusing" instances from the training data, thereby reducing the size of the learned decision trees.

Table 3: Tree size – number of leaves

Noise	None	CF
0	187.7	121.4
5	270.7	126.0
10	333.0	143.5
20	419.2	189.7
30	484.6	262.4
40	517.7	302.8

The reduction in tree size and the improvement in accuracy raise the question as to whether filtering is just getting rid of those parts of the data that are difficult to classify, thereby achieving higher overall accuracy. In other words, is the filtering procedure throwing out the instances of a particular class, or of a subset of the classes, to achieve higher accuracy on the remaining classes. To test this hypothesis we examined the decision tree's accuracy for each class at each noise level. The results are shown in Table 4. We have placed a † by the class numbers of those classes that were artificially corrupted.

For a noise level of 5% filtering decreased accuracy for classes 3, 5, 7 and 9, but by less than 2.5%. For a noise level of 10% the accuracy of class 1 went down by 0.7 %. For a noise level of 20% classes 5 and 10 decreased in accuracy. For a noise level of 30% the accuracies of classes 5 and 7 decreased with filtering and for 40% the accuracy of classes 4, 5, 6, and 8 decreased. For classes that did not have noise added to them (classes 1 and 9), filtering generally did not decrease accuracy and in some cases increased accuracy. For class 5, filtering decreases accuracy in four out of the five cases. Class 5 is unique in that its labels were artificially corrupted with both class 10 and class 2, thereby doubling the noise level. In general, the results show that for this dataset, filtering does not sacrifice the accuracy of some of the classes to improve overall

Table 4: Decision tree accuracy by class

Noise	Filter	1	2 †	3 †	4 †	5 †	6 †	7 †	8 †	9	10 †	11 †
5	None	94.3	78.4	92.7	92.9	40.0	81.0	66.8	90.4	78.5	0.0	82.0
5	CF	96.0	79.7	91.8	95.6	38.0	84.8	64.4	97.3	77.7	0.0	88.0
10	None	96.0	73.1	80.9	86.7	46.0	74.8	58.2	89.7	73.5	0.2	74.0
10	CF	95.3	79.7	88.2	94.8	46.0	83.8	63.2	94.2	78.4	0.2	81.3
20	None	94.0	64.4	65.4	80.0	40.0	68.1	55.9	74.5	76.3	20.0	70.0
20	CF	96.6	75.9	80.0	87.7	38.0	78.6	62.1	89.7	77.9	0.0	72.7
30	None	95.3	50.9	54.5	64.4	22.0	58.1	52.4	67.9	74.2	0.0	54.0
30	CF	95.7	64.1	54.5	76.3	16.0	68.1	50.9	74.8	81.5	0.0	71.3
40	None	94.2	44.4	47.3	47.0	18.0	47.1	32.4	44.8	72.7	0.0	39.4
40	CF	97.0	51.2	49.1	45.1	12.0	43.8	35.3	42.8	83.8	0.0	47.3

accuracy, and that it can retain base-line accuracy in noisy classes for noise levels of up to 20%.

Filter Precision

To assess the consensus filter's ability to identify mislabeled instances, we examined the intersection between the set of instances that were corrupted and the set of instances that were tagged as mislabeled by the consensus filter. In Figure 1 this is the area $M \cap D$. The results of this analysis are shown in Table 5. Each row in the table reports the average over the ten runs of the number of instances discarded by the filter $\mid D \mid$, corrupted in the data $\mid M \mid$, in both sets $\mid M \cap D \mid$, and estimates of the probability of making an E_1 or an E_2 error. $P(E_1)$ represents the probability of throwing out good data and can be estimated as:

$$P(E_1) = \frac{Discarded - Intersect}{Total - Corrupted} = \frac{\mid D \mid - \mid M \cap D \mid}{Total - \mid M \mid}$$

$P(E_2)$ represents the probability of keeping bad data and can be estimated as:

$$P(E_2) = \frac{Corrupted - Intersect}{Corrupted} = \frac{\mid M \mid - \mid M \cap D \mid}{\mid M \mid}$$

There are 3063 (90% of 3398) total training instances. Therefore, for a noise level of 5%, $P(E_1) = \frac{257.8-89.5}{3063-103.0} = .057$ and $P(E_2) = \frac{103.0-89.5}{103.0} = .131$.

The results show that the probability of throwing out good data remains small even for higher noise levels, illustrating that the consensus filter is conservative in discarding data. On the other hand, the results illustrate that the probability of keeping bad data grows rapidly as the noise level increases. Indeed for a noise level of 40% it has a 72% chance of retaining mislabeled instances. This comes as no surprise since 40% noise makes it difficult to distinguish between pairs of classes that have been corrupted, which is evident from the low accuracies observed for these classes in Table 4.

Table 5: Consensus filter precision

	Number of Instances			Prob. of Error	
	$\mid D \mid$	$\mid M \mid$	$\mid M \cap D \mid$	$P(E_1)$	$P(E_2)$
5	257.8	103.0	89.5	0.057	0.131
10	353.7	217.7	171.8	0.064	0.211
20	465.0	422.7	272.8	0.073	0.355
30	559.8	712.2	324.9	0.100	0.544
40	609.8	1106.4	314.8	0.151	0.716

For higher levels of noise, the consensus filter did not find many of the mislabeled instances. For domains with high class noise a less conservative approach may do a better job at minimizing type E_2 errors. Indeed, a drawback of the consensus filter is that it minimizes E_1 errors at the expense of incurring more E_2 errors. Therefore future work will focus on modeling this tradeoff explicitly as a parameter. In addition, we will develop methods for customizing this parameter to the dataset characteristics of the particular task at hand. A straightforward way to model the tradeoff is to have the parameter be set to be the minimum number of base-level classifiers that must label an instance as noisy before it can be discarded. At one end of the spectrum all base-level classifiers must label an instance and at the other only one must label an instance before it can be discarded.

Conclusions and Future Directions

This article presented a procedure for identifying mislabeled instances. The results of an empirical evaluation demonstrated that the consensus filter method improves classification accuracy for a land-cover mapping task for which the training data contains mislabeled instances. Filtering allowed the base-line accuracy to be

retained for noise levels up to 20%. An evaluation of the precision of the approach illustrated that consensus filters are conservative in throwing away good data, at the expense of keeping mislabeled data.

A future direction of research will be to extend the filter approach to *correct* labeling errors in training data. For example, one way to do this might be to relabel instances if the consensus class is different than the observed class. Instances for which the consensus filter predicts two or more classes would still be discarded. This direction is particularly important because of the paucity of high quality training data available for many applications.

Finally, the issue of determining whether or not to use the consensus filter method for a given data set must be considered. For the work described here, the data were artificially corrupted. Therefore the nature and magnitude of the labeling errors were known a priori. Unfortunately, this type of information is rarely known for most "real world" applications. In some situations, it may be possible to use domain knowledge to estimate the amount of label noise in a dataset. For situations where this knowledge is not available, the conservative nature of our filtering procedure dictates that relatively few instances will be discarded for data sets with low levels of labeling error (see Table 5). Therefore, the application of this method to relatively noise free datasets should not significantly impact the performance of the final classification procedure.

Acknowledgments

We would like to thank our reviewers for their careful and detailed comments and Ruth Defries for supplying the data used for this work.

References

Aha, D., Kibler, D., & Albert, M. (1991). Instance-based learning algorithms. *Machine Learning, 6*, 37-66.

Benediktsson, J., & Swain, P. (1992). Consensus theoretic classification methods. *IEEE Transactions on Systems, Man, and Cybernetics, 22*, 668-704.

Brodley, C. E., & Utgoff, P. E. (1995). Multivariate decision trees. *Machine Learning, 19*, 45-77.

Cardie, C. (1993). Using decision trees to improve case-based learning. *Machine Learning: Proceedings of the Tenth International Conference* (pp. 25-32). Amherst, MA: Morgan Kaufmann.

Danyluk, A., & Provost, F. (1993). Small disjuncts in action: Learning to diagnose errors in the telephone network local loop. *Machine Learning: Proceedings of the Tenth International Conference* (pp. 81-88). Amherst, MA: Morgan Kaufmann.

Defries, R. S. , & Townsend, J. R. G. (1994). NDVI-derived land cover classifications at a global scale. *International Journal of Remote Sensing, 15*, 3567-3586.

Hansen, L. K., & Salamon, P. (1990). Neural network ensembles. *IEEE Transactions of Pattern Analysis and Machine Intelligence, 12*, 993-1001.

John, G. H. (1995). Robust decision trees: Removing outliers from data. *Proceedings of the First International Conference on Knowledge Discovery and Data Mining* (pp. 174-179). Montreal, Quebec: AAAI Press.

Lewis, D., & Catlett, J. (1994). Heterogeneous uncertainty sampling for supervised learning. *Machine Learning: Proceedings of the Eleventh International Conference* (pp. 148-156). New Brunswick, NJ: Morgan Kaufmann.

Nilsson, N. J. (1965). *Learning machines.* New York: McGraw-Hill.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning, 1*, 81-106.

Quinlan, J. R. (1993). *C4.5: Programs for machine learning.* San Mateo, CA: Morgan Kaufmann.

Skalak, D., & Rissland, E. (1990). Inductive learning in a mixed paradigm setting. *Proceedings of the Eighth National Conference on Artificial Intelligence* (pp. 840-847). Boston, MA: Morgan Kaufmann.

Skalak, D. (1994). Prototype and feature selection by sampling and random mutation hill climbing algorithms. *Machine Learning: Proceedings of the Eleventh International Conference* (pp. 293-301). New Brunswick, NJ: Morgan Kaufmann.

Weisberg, S. (1985). *Applied linear regression.* John Wiley & Sons.

Wilson, D. (1972). Asymptotic properties of nearest neighbor rules using edited data. *IEEE Trans. on Systems, Man and Cybernetics, 2*, 408-421.

Winston, P. H. (1975). Learning structural descriptions from examples. In Winston (Ed.), *The psychology of computer vision.* New York: McGraw-Hill.

Wolpert, D. H. (1992). Stacked generalization. *Neural Networks, 5*, 241-259.