Proceedings of the Fourteenth International Conference on Machine Learning, 1997, Morgan
Kaufmann, pages 57–65.

Improving Minority Class Prediction Using Case-Specific Feature Weights

Claire Cardie
Department of Computer Science
Cornell University
Ithaca, NY 14850–7501
cardie@cs.cornell.edu

Nicholas Howe
Department of Computer Science
Cornell University
Ithaca, NY 14850–7501
nihowe@cs.cornell.edu

Abstract

This paper addresses the problem of handling skewed class distributions within the case-based learning (CBL) framework. We first present as a baseline an information-gain-weighted CBL algorithm and apply it to three data sets from natural language processing (NLP) with skewed class distributions. Although overall performance of the baseline CBL algorithm is good, we show that the algorithm exhibits poor performance on minority class instances. We then present two CBL algorithms designed to improve the performance of minority class predictions. Each variation creates test-case-specific feature weights by first observing the path taken by the test case in a decision tree created for the learning task, and then using path-specific information gain values to create an appropriate weight vector for use during case retrieval. When applied to the NLP data sets, the algorithms are shown to significantly increase the accuracy of minority class predictions while maintaining or improving overall classification accuracy.

1 INTRODUCTION

Many real-world machine learning tasks exhibit skewed class distributions in that a small number of classes (often a single class) cover the majority of the instance population (Fawcett, 1996). Data sets in medical diagnosis domains, for example, typically contain many "negative/no disease" instances, but few instances that correspond to positive diagnoses for the disease. One example is the thyroid data set used in Aha & Kibler (1987) where negative cases comprise 95% of the available instances. Skewed class distributions are also the norm for learning problems in natural language processing (NLP). McCarthy's two-class coreference resolution data sets contain 20-25% positive examples (McCarthy and Lehnert, 1995); Soderland's discourse analysis data sets generally contain only 5% positive instances (Soderland and Lehnert, 1994). We have seen similar trends for multi-class NLP problems: in our part-of-speech tagging data (11 classes), two classes account for 81.5% of the data; for semantic class tagging (34 classes), one class accounts for 58.1% of the data; for concept extraction (10 classes), one class accounts for 91.7% of the instances (Cardie, 1993a).

Not surprisingly, when trained on such skewed distributions, most machine learning algorithms exhibit accurate prediction for *majority class instances* — instances associated with the prevalent class(es), but exhibit very poor performance for *minority class instances* — instances associated with any of the remaining, low frequency classes. Unfortunately, achieving reasonable accuracy on minority class instances is often of utmost importance. In the case of concept extraction in NLP, for example, the task for the learning algorithm is to decide when and which information should be extracted from an input text and included in its summary. Because such concept extraction situations occur relatively infrequently, a default decision *never* to extract concepts as the parser proceeds through the text achieves high accuracy (91.7% correct). Nevertheless, an NLP system that adopts this default strategy would just produce an empty summary for each input text. Accurate prediction of minority classes is important in a variety of other domains as well: failing to detect a disease in medical diagnosis domains can be disastrous, as can failing to predict large, but rare, declines in the stock market.

These domains, in effect, measure performance by associating a greater error cost for misclassifications of minority class values than of majority class values. In spite of its prevalence in real-world domains, however, little research to date has focused on learning with skewed class distributions (Fawcett, 1996).

This paper addresses the problem of learning with skewed class distributions within the case-based learning (CBL) framework. We first present as a baseline an information-gain-weighted case-based learning algorithm, IG-CBL (Bosch and Daelemans, 1993; Cardie, 1993b). IG-CBL is a weighted k-nearest neighbor (k-nn) algorithm that derives feature weights based on the information gain of the associated feature across the training data. We then apply the algorithm to three problems from NLP with skewed class distributions: part-of-speech tagging, semantic class tagging, and the acquisition of information extraction patterns, i.e., concept extraction. Although overall performance of the baseline IG-CBL algorithm is good, we show that the algorithm exhibits poor performance on minority class instances and hypothesize that one problem with the baseline algorithm is its globally computed weight vector: a single set of feature weights is computed for each classification task and used in the case retrieval distance metric that determines case similarity throughout testing. We next present two CBL algorithms, Case-Specific-IG and Combo-IG, that are designed to improve the performance of minority class predictions. In contrast to the baseline, each algorithm creates *test-case-specific feature weights* by (1) observing the path taken by the test case in a decision tree created for the learning task, and then (2) using path-specific information gain values to create an appropriate weight vector for use during case retrieval. When applied to the NLP data sets, the algorithms are shown to significantly increase the accuracy of minority class predictions — from 70.6% to 80.6% for part-of-speech tagging; from 59.6% to 67.1% for semantic class tagging; and from 56.4% to 64.6% for concept extraction. In addition, overall classification accuracy is maintained or improved.

In the remainder of the paper we first present the baseline CBL algorithm and demonstrate its performance on the three NLP data sets described above (Section 2). We then introduce our approach for creating case-specific feature weights and evaluate its performance on the three data sets (Section 3). Section 4 provides an analysis of the results. Related work is discussed in Section 5.

2 THE BASELINE CBL ALGORITHM: IG-CBL

This section presents and evaluates IG-CBL, a realistic baseline for the NLP data sets used throughout the paper. In particular, we wanted a baseline that performs as well as or better than previous studies using these and other NLP data sets (e.g., Cardie(1993a), Daelemans(1996a)). As a result, we use IG-CBL, a weighted k-nearest neighbor case-based learning algorithm modified to handle the entirely symbolic-valued feature sets of the NLP tasks. IG-CBL's feature weighting algorithm is a straightforward composition of two existing approaches:

Feature selection. IG-CBL first uses a decision tree for feature selection as described in Cardie(1993b) and briefly below. The goal in this step is to prune features from the representation so that the CBL algorithm can ignore them entirely.

Feature weighting. IG-CBL then assigns each remaining feature a weight according to its information gain across the training cases (Bosch and Daelemans, 1993; Daelemans *et al.*, 1997). The intent here is to weight each feature relative to its overall importance in the data set.

Training Phase. There are three steps to the IG-CBL training phase:

1. *Create the case base.* For this, we simply store all of the training instances in a flat case base.
2. *Use the training instances to create a decision tree for the learning task.* We are using a Common Lisp implementation[1] of C4.5 (Quinlan, 1992). Because features in the NLP data set have a widely varying number of possible values, we use the information gain ratio as a splitting criterion.
3. *Compute feature weights for use during case retrieval.* For each feature, f, we compute a weight, w_f, as follows:

   $$\begin{array}{ll} w_f = G(f) & \text{if } f \text{ is in the tree of step 2;} \\ w_f = 0 & \text{otherwise;} \end{array}$$

   where $G(f)$ is the information gain of f as computed across all training instances by C4.5.

Testing/Application Phase. After training, the class value for a novel instance, X, is determined as follows:

[1] We thank Joe McCarthy and UMass, Amherst for this version of C4.5.

1. Compare the test case, X, to each case, Y, in the case base and calculate, for each pair:

   $$\sum_{i=1}^{|F|} w_{f_i} * match(X_{f_i}, Y_{f_i})$$

   where F is the feature set, w_{f_i} is the weight of feature i as determined during training, and X_{f_i} and Y_{f_i} are the values of feature i in X and Y, respectively. $match(a, b)$ is a function that returns 1 if a and b are equal; 0.5 if a and b are partial matches[2]; and 0 otherwise.
2. Return the k cases with the highest-score as well as any ties.
3. Let the retrieved cases vote on the class value of X. In the case of ties, choose randomly.[3]

2.1 EVALUATION OF THE IG-CBL BASELINE

To evaluate the IG-CBL baseline algorithm, we applied it to the part-of-speech (p-o-s), semantic class (sem class), and concept extraction (concept) data sets, which are described in detail in Cardie(1993a). Each data set contains 2056 cases. The cases were created automatically by the CIRCUS parser (Lehnert, 1990; Cardie and Lehnert, 1991) and represent the context in which CIRCUS encounters each content word in its left-to-right traversal of 120 randomly selected sentences from the MUC business joint ventures corpus (MUC-5, 1994). At a minimum, the parser must decide the part of speech, semantic class, and concept extraction information for each of these content words.[4] Each case comprises 33 features. Twenty-two of these describe the local context in which the test word was encountered (i.e., the linguistic features of the words in a 5-word window centered on the test word). The remaining 11 global context features of each case encode information for any major syntactic constituents that have been recognized in the current clause at the time that the test word is encountered. All experiments use 10-fold cross validation at the sentence level. In addition, the same folds were used across all experiments throughout the paper.

Table 1 shows the results of applying IG-CBL with $k = 10$ to the three NLP datasets and compares its performance with the C4.5 decision tree algorithm and with a default strategy that always chooses the most frequent class value (see columns three through five).[5] The number of class values and minority class values for each data set is shown in column two of the table. Chi-square significance tests indicate that IG-CBL performs significantly better than both alternatives ($p = .01$) in all but one case: the performance of the decision tree and IG-CBL are indistinguishable for the concept extraction task. While the overall performance of IG-CBL is quite good[6], the last column shows that the algorithm's performance on minority class instances is less than stellar. For the three data sets, IG-CBL's accuracy across *all* classes is superior to its accuracy across just the minority classes: 93.7% correct (across all classes) vs. 70.6% correct (minority classes) for part-of-speech tagging; 79.9% vs. 59.6% for semantic class tagging; and 94.5% vs. 56.4% for concept extraction. Drops in performance also occur when measuring minority class performance for the decision tree algorithm.

One option for improving minority class prediction is suggested by Wettschereck (1994). He finds that smaller values of k should be employed when the number of training instances available for a class is small, as is often the case for minority class instances in skewed instance populations. In our data, for example, the average number of instances for each minority class is 42.3 (p-o-s), 26.1 (sem class), and 19.0 (concept) vs. 837.5, 1195.0, and 1885.0 for the majority class(es), respectively. Unfortunately, using $k = 1$ or $k = 5$ rather than $k = 10$ does not help here: for part-of-speech tagging, the minority class performance of IG-CBL at $k = 1$ drops from 70.6% to 69.5%; for semantic class tagging, performance drops from 59.6% to 57.9%; and for concept extraction, performance increases slightly from 56.4% to 56.7%. None of these changes is statistically significant.

We surmise instead that IG-CBL's global feature weighting is responsible for the algorithm's poor performance on minority class instances — IG-CBL creates a single weight vector for each classification task. This weight vector is based on all of the training cases.

[2] Some of the features in the NLP data sets can be lists of values. A partial match is a non-nil intersection of such lists.

[3] For ties, we actually return the first class value in the list of values returned.

[4] Note that CIRCUS has available a small lexicon of function words (e.g., the, a, an, is, are, and, or). We are using the machine learning algorithm only to determine information for non-function words and for a few highly ambiguous function words.

[5] We also tested the algorithm with $k = 1$ and $k = 5$, but better results were obtained with $k = 10$ as was the case in Cardie(1993a,1993b). Furthermore, we tested a k-nn algorithm that uses *all* available features (i.e., $w_f = 1$) as well as the simpler 0-1 feature weighting scheme of Cardie(1993b). The "all features" versions perform poorly; the 0-1 weighting scheme consistently performs about five percentage points worse than IG-CBL for each data set.

[6] Note that the table shows results for *content* words only. As expected, overall tagging results improve when function words (e.g., the, a, an, is, are, and, or) are included. This is because CIRCUS's lexicon has special procedures that accurately handle most function words.

Table 1: Baseline Results for IG-CBL. % indicates percentage correct.

Data Set	# classes/ # minority classes	Overall Accuracy			Minority Class Accuracy	
		Default	C4.5	IG-CBL	C4.5	IG-CBL
p-o-s	11/9	81.5%	92.2%	93.7%	68.2%	70.6%
sem class	34/33	58.1%	63.9%	79.9%	16.2%	59.6%
concept	10/9	91.7%	94.6%	94.5%	55.3%	56.4%

As a result, for data sets where skewed class distributions exist, the algorithm is biased towards choosing weights that improve prediction of majority class instances. We hypothesize, therefore, that performance on minority class data can be recovered by replacing IG-CBL's global weight vector with a set of dynamically generated weight vectors that combine the advantages of information gain weighting with the benefits of a finer-grained matching during case retrieval.[7] In particular, we rely on the fact that only a portion of the decision tree created for the classification task is relevant to a test instance — namely, the single path from the root to a leaf that would be traversed by the test case. In the next section, we show how such test-case-specific weight vectors can be created.

3 TEST-CASE-SPECIFIC FEATURE WEIGHTING

In an attempt to improve minority class predictions, this section presents two related algorithms for creating test-case-specific feature weights: Case-Specific-IG and Combo-IG. Given a test case, each variation produces a weight vector by first observing the path taken by the test case in a decision tree created for the learning task, and then using path-specific information gain values to create an appropriate weight vector for use during case retrieval.

As described above, we hypothesize that this should produce feature weights tailored to the specific test case and allow better matches for minority class instances during case retrieval while preserving the accuracy on majority classes. In particular, the Case-Specific-IG algorithm creates a weight vector, w_X, for test case X that (a) ignores all features not on X's path p through the decision tree, and (b) weights the remaining features according to their relative positions along p. The Combo-IG algorithm, on the other hand, combines the case-specific weights with the global weight vector used in the baseline IG-CBL algorithm. Each algorithm is described in detail and evaluated below.

3.1 CASE-SPECIFIC IG WEIGHTING (CASE-SPECIFIC-IG)

The training phase for the Case-Specific-IG algorithm is identical to that of the IG-CBL baseline except that no feature weights are computed. Instead, the weights are computed dynamically as each test case is encountered. Like IG-CBL, we use the decision tree for feature selection as well as for feature weighting. Now, however, the features selected for use in case retrieval are those that appear as decision criteria along the path followed in the decision tree for the current test case. The objective of Case-Specific-IG's weighting scheme is simple: each feature along this path, p, should be assigned a weight greater than that of any feature tested lower in the tree along p. We are exploring a variety of methods to create such a weight vector and have found a number of them to work well. The experiments below, however, calculate weights according to the following scheme: Let $p = n_1, n_2, \ldots, n_l$ be the path taken by the test case through the decision tree where n_1 refers to the root, n_2 refers to the next node in the path, ..., and n_l refers to the leaf where the path terminates. Then the weight of the root attribute in the decision tree, f_{n_1}, is the information gain associated with knowing the values of all the attributes along p; the weight of f_{n_2}, is the information gain associated with knowing the values of attributes f_{n_1} through $f_{n_{l-1}}$; etc. In general, the weight of f_{n_i}, is the information gain associated with knowing the values of attributes f_{n_1} through $f_{n_{l-i+1}}$.[8] More specif-

[7] An intermediate step between task-based feature weights and test-case-specific feature weights would be to use information gain to compute a weight vector for each class value. During testing, all training instances with the same class value would be assigned the weight vector associated with that class value. We explore one method for computing class-specific weights in Howe & Cardie (1997).

[8] A possibly more intuitive approach would let the weight of f_{n_i} be the information gain associated with knowing the values of attributes f_{n_i} through f_{n_l}. This works less well than the method described here. We believe that this is because it biases retrieval towards cases that match features towards the top of the tree (which are probably important for majority class prediction) rather than features that appear lower in the tree (which may be useful for minority class prediction).

ically, the following algorithm is used to create w_X, the weight vector for test case X:

1. Present X to the decision tree for classification and note the path, $p = n_1, n_2, \ldots, n_l$ that is taken through the tree.
2. Assign a weight of 0 to any feature that does *not* appear along path p.
3. Next calculate weights $w_{f_{n_k}}$ for features that appear along p. In the equations below, f_{n_i} refers to the feature tested at node n_i; T_{v_i} stands for the subset of cases in T that have the value v_i for the corresponding feature f_{n_i}; $T_{v_1 \ldots v_j}$ is the subset of cases that have values $v_1 \ldots v_j$ for the corresponding features $f_{n_1} \ldots f_{n_j}$; and F_i stands for the set of values taken on by feature f_i in T. Then,

   $$w_{f_{n_k}} = H(T) - H_{f_{n_1} \cdots f_{n_{l+1-k}}}(T),$$
   $$for\ f_{n_k} \in \{f_{n_1}, \ldots, f_{n_l}\}$$

   where $H(T)$ is the entropy in the training set T:

   $$H(T) = - \sum_{c \in Classes} \frac{|T_c|}{|T|} log_2 \frac{|T_c|}{|T|}$$

   and $H_{f_{n_1} \cdots f_{n_j}}$ is the entropy in T when the values of features f_{n_1} to f_{n_j} are known as calculated according to standard information theory:

   $$H_{f_{n_1} \cdots f_{n_j}}(T) = \sum_{v_1 \in F_1} \cdots \sum_{v_j \in F_j} \frac{|T_{v_1 \ldots v_j}|}{|T|} H(T_{v_1 \ldots v_j}).$$

Once the weight vector has been computed, we determine the class of X by invoking the k-nn case retrieval algorithm of IG-CBL with $k = 1$. A 1-nn algorithm is used here because the primary goal of Case-Specific-IG is to improve the performance of low-frequency, minority class instances for which Wettschereck (1994) suggested small values of k. Larger values of k were tested, but did not perform as well.

3.2 COMBINING THE CASE-SPECIFIC AND GLOBAL IG WEIGHTS

This section describes Combo-IG, a variation of Case-Specific-IG that attempts to balance performance for minority and majority classes by combining the test-case-specific feature weights of Case-Specific-IG with the globally computed weight vector of the baseline IG-CBL algorithm. The training phase for the Combo-IG algorithm is identical to that of IG-CBL: (1) store the training instances in the case base, (2) train a decision tree using the same set of training instances, and (3) compute the weight vector used in IG-CBL, which we will call *global-IG*. During testing, the case-specific feature weights, w_X, for test case X are computed as follows:

$$w_X = \textit{global-IG} + \textit{CS-IG}_X$$

where *CS-IG*$_X$ is the test-case-specific weight vector of Case-Specific-IG. As usual, Combo-IG determines the class of X using w_X and the k-nn case retrieval algorithm of IG-CBL with $k = 1$.

In general, all algorithms presented in this paper represent a collection of feature-weighting algorithms, each of which creates a weight vector of the following form:

$$w_X = \lambda_1\ (\textit{global-IG}) + \lambda_2\ (\textit{CS-IG}_X).$$

For IG-CBL, $\lambda_1 = 1$ and $\lambda_2 = 0$; for Case-Specific-IG, $\lambda_1 = 0$ and $\lambda_2 = 1$; and for Combo-IG, $\lambda_1 = 1$ and $\lambda_2 = 1$. In the next section, we evaluate the performance of Case-Specific-IG and Combo-IG by applying them to the part-of-speech, semantic class, and concept extraction data sets.

3.3 A COMPARATIVE EVALUATION

Table 2 compares the minority class performance of the case-specific weighting algorithms (Case-Specific-IG and Combo-IG) to the baseline IG-CBL algorithm. The final column in the table presents a variation of Combo-IG in which the case-specific weights are doubled before being added to the global-IG weights (i.e., $\lambda_1 = 1$ and $\lambda_2 = 2$) in order to emphasize the case-specific weights. For all data sets tested, at least two of the test-case-specific variations produce significant improvements ($p \leq .05$) for prediction of minority class instances. In addition, only one of nine variations is clearly worse. The Case-Specific-IG algorithm produces the best results for the part-of-speech and concept extraction data sets. Adding the global-IG weights to the Case-Specific-IG weights for these data sets only degrades performance, although the dips in performance produce accuracies that are still significantly greater than those of the baseline. The semantic class data set behaves differently. For this data set, the Case-Specific-IG weights drastically reduce the ability of the CBL algorithm to correctly predict minority classes; the Combo-IG algorithms, however, perform significantly better than the baseline ($p \leq .05$ for Combo-IG; $p \leq .10$ for Combo-IG-2x).

Table 3 provides a closer look at specific changes in the prediction of individual class values when using case-specific feature weighting vs. the global weight vector of IG-CBL. The table focuses only on part-of-speech tagging and shows changes across the two majority classes (noun and noun modifier) as well as the minority classes.

As stated above, the our goal for the feature weighting algorithms was to maintain or improve overall accuracy while recovering performance on minority classes. Table 4, therefore, examines the effects of

Table 2: Minority Class Performance of the Case-Specific Weighting Algorithms. % indicates percentage correct. The symbols **, *, and + indicate significance with respect to the baseline IG-CBL results: ** → $p = .01$, * → $p = .05$, and + → $p = .10$.

Data Set	# minority instances	IG-CBL	Case-Specific-IG	Combo-IG	Combo-IG-2x
p-o-s	381	70.6%	80.6%**	76.9%**	79.3%**
sem class	861	59.6%	49.4%**	65.5%**	67.1%**
concept	171	56.4%	64.6%*	63.7%*	63.2%+

Table 3: Detailed Results for Part-of-Speech Tagging.

Class	# of Instances	IG-CBL % correct (# correct)	Case-Specific-IG % correct (# correct)
modal	1	0 (0)	0 (0)
connective	2	0 (0)	0 (0)
preposition	3	0 (0)	0 (0)
gerund	5	40.0 (2)	60.0 (3)
present participle	15	26.7 (4)	66.7 (10)
verb particle	24	79.2 (19)	83.3 (20)
past participle	73	71.2 (52)	87.7 (64)
adverb	83	54.2 (45)	69.9 (58)
verb	175	84.0 (147)	86.9 (152)
noun	808	98.8 (798)	97.2 (785)
noun modifier	867	99.2 (860)	96.5 (837)

case-specific feature weighting on overall accuracy. It shows that overall accuracy is, in fact, maintained for part-of-speech tagging. For the semantic class data set, overall accuracy is maintained only when both the case-specific and the global weights are used for case retrieval. For concept extraction, the case-specific weights alone significantly improve overall accuracy, while the combined weight vectors significantly degrade accuracy. Remember, however, that the figures indicate accuracy across all classes: For skewed class distributions, overall utility is best measured using an error cost matrix that establishes higher penalties for errors on minority class values. For many such loss functions, all of the proposed feature-weighting algorithms would maintain or improve overall classification performance.

4 DISCUSSION OF RESULTS

The last section showed that test-case-specific feature weights based on information gain can improve the performance of a case-based learning algorithm on minority class data while maintaining or improving overall classification accuracy. However, there remain a number of problems with the approach.

In the experiments above, at least two of the three proposed feature weighting methods were successful for each data set, but without further experimentation on additional learning tasks, we cannot predict, a priori, which method will work best on a particular data set. Given just the results on the three NLP data sets, however, it seems that for data sets where the Case-Specific-IG weights alone improve minority class performance, that method will outperform the proposed alternatives. For data sets where the Case-Specific-IG variation degrades minority class performance, combining the case-specific and global-IG weights will be the better option. It is likely that the method of choice depends on the number of minority cases available and the distribution of those instances across the minority classes.

In addition, the test-case-specific feature weighting approaches presented here were not designed to handle all tasks with skewed class distributions. In particular, we are unable to achieve significant increases in overall accuracies for two of the three data sets. The method is designed more for problems in which the misclassification cost for errors on minority classes is higher than for majority classes.

Recent work by Daelemans *et al.*(1996) suggests that, in general, our approach will work well for data sets in which there is a noticeable variation in the infor-

Table 4: Overall Performance of the Case-Specific Weighting Algorithms (% correct). $k = 10$ is used for the baseline runs; $k = 1$ elsewhere. The symbols **, *, and + indicate significance with respect to the baseline IG-CBL results: ** $\rightarrow p = .01$, * $\rightarrow p = .05$, and + $\rightarrow p = .10$.

Data Set	IG-CBL Baseline	Case-Specific IG	Combo-IG	Combo-IG-2x
p-o-s	93.7	93.8	93.7	94.1
sem class	79.9	77.0**	79.1	79.8
concept	94.5	95.7**	93.1**	93.1**

mation gain values of the features. Their work introduces IGTree, an algorithm that uses information gain to compress a case base into a tree structure while allowing efficient case retrieval and maintaining good generalization capabilities. As part of their investigation, they determined that both IGTree and an algorithm very similar to the IG-CBL baseline perform worse than an unweighted k-nearest neighbor algorithm when the mutual differences in information gain values of the features is small. We expect the same to hold for the case-specific IG-weighted algorithms presented here.

5 RELATED WORK AND CONCLUSIONS

While little research in the machine learning community has focused explicitly on improving the performance of minority class instances, a variety of existing methods implicitly address this problem. For example, learning algorithms that allow the user to define misclassification costs (e.g., Breiman (1984), Pazzani (1994)) can be used to improve minority class performance by defining a cost matrix (or vector) that assigns minority class errors a greater penalty than majority class errors. One possible problem with these approaches, however, is that they are not specifically designed to learn with skewed class distributions and may not perform well under those conditions. Lewis & Catlett (1994), on the other hand, introduce a cost parameter to C4.5 (i.e., the *loss ratio*) specifically to handle skewed distributions. Extending their approach to the multi-class case would allow its application to the NLP data sets used here.

Fisher's Cobweb system (1987) also has some similarities to the methods introduced here. Cobweb creates a multivariate classification tree with training cases at the terminal nodes, and bases most predictions on these stored cases. Because the system uses probabilistic weights to sort test cases, it effectively assigns different weights to each path it its tree, achieving an effect similar to ours.

Freund & Schapire's AdaBoost (1996) is yet another alternative for implicitly dealing with the problem of skewed class distributions. AdaBoost successively builds a series of classifiers, each of which is designed to better handle the errors of the preceding set. Classifying a novel instance proceeds by combining the predictions of the individual classifiers. For problems where minority class prediction is poor, AdaBoost should automatically devote energy to improving performance on minority class instances as it creates new classifiers. In contrast to the methods presented here, AdaBoost improves overall performance by choosing the appropriate training instances for each classifier rather than by selecting and weighting features appropriately. To perform well in domains with many irrelevant features (e.g., the NLP tasks tested here), however, AdaBoost would require additional feature selection and feature weighting methods.

Within the arena of case-based learning algorithms, Aha & Kibler's IB3 (1991) has characteristics that should enable it to deal effectively with skewed distributions. The approach is quite different from the one presented here: IB3 uses a combination of the class frequency and predictive accuracy of individual training instances to decide whether the instance should be stored in the case base and used to classify subsequent cases.

Our approach to handling skewed class distributions is an example of a growing collection of local feature weighting schemes for case-based learning algorithms. (For an excellent survey and empirical analysis of feature-weighting methods for k-nearest neighbor algorithms, see Wettschereck *et al.* (1997).) Local feature weighting algorithms allow feature weights to vary across the instance space. Some local weighting methods (e.g., the value difference metric of Stanfill & Waltz (1986)) assign a different weight to each value of a feature. Others associate a different weight vector with every training case rather than with every test case as is done here. Aha & Goldstone (1992), for example, present one such training-case-specific weighting scheme that has some similarities to ours: they compute a weight vector for each training case by combining globally and locally computed feature weights. The greater the similarity of a test case to the training case, the greater the emphasis of the training case weights over the global weights. Another algorithm that allows feature relevance to vary across the training instances is the RC algorithm of Domingos (1997). This algorithm uses a context-sensitive clustering method to perform feature selection rather than to assign continuous feature weights. Most similar to our approach, however, are methods that create what Wettschereck *et al.*(1997) call *query-specific* weights. Atkeson *et al.*(1997), for example, create a different similarity metric for each test case, but do so in a (continuous) function-learning paradigm rather than a classification paradigm. In addition, Hastie & Tibshirani (1994) and Friedman (1994) compute test-case-specific metrics that rely on discriminant analysis and recursive partitioning, respectively.

In summary, we have investigated the use of test-case-specific feature weighting to aid in the recovery of minority class instances in skewed class distributions. We presented two case-based learning algorithms that use decision trees to create the case-specific weight vectors. Each variation composes the vector in two stages. In a feature selection stage, the path that would be taken by the test case in a decision tree created for the learning task is noted. Any feature tested along the path is included in the case representation; all other features are ignored. Weights for the selected features are then determined using path-specific information gain values. On three data sets with skewed distributions from the natural language processing domain, the algorithms are shown to significantly increase the accuracy of minority class predictions while maintaining or improving overall classification accuracy. Given our initial results, we believe that this is a promising approach for dealing with skewed distributions in other domains.

6 ACKNOWLEDGMENTS

We thank David Skalak for his advice and suggestions on this work. We also thank the anonymous reviewers for their helpful comments. This work was supported in part by NSF CAREER Award IRI–9624639 and NSF Grant GER–9454149.

References

(Aha and Goldstone, 1992) Aha, D. W. and Goldstone, R. L. 1992. Concept learning and flexible weighting. In *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society*, Indiana Univeristy, Bloomington, IN. Lawrence Erlbaum Associates. 534–539.

(Aha and Kibler, 1987) Aha, D. and Kibler, D. 1987. Learning Representative Exemplars of Concepts: An Initial Case Study. In *Proceedings of the Fourth International Conference on Machine Learning*, U. C. Irvine, Irvine, CA. Morgan Kaufmann. 24–30.

(Aha *et al.*, 1991) Aha, D.; Kibler, D.; and Albert, M. 1991. Instance-Based Learning Algorithms. *Machine Learning* 6(1):37–66.

(Atkeson *et al.*, 1997) Atkeson, C. G.; Moore, A. W.; and Schaal, S. 1997. Locally weighted learning. *Artificial Intelligence Review* 11:11–73.

(Bosch and Daelemans, 1993) Bosch, A. van den and Daelemans, W. 1993. Data-oriented methods for grapheme-to-phoneme conversion. In *Proceedings of the Sixth Conference of the EACL*, Utrecht. 45–53. Also available as ITK Research Report 42.

(Breiman *et al.*, 1984) Breiman, L.; Friedman, J.H.; Olshen, R.A.; and Stone, C.J. 1984. *Classification and Regression Trees.* Wadsworth International Group, Belmont, CA.

(Cardie and Lehnert, 1991) Cardie, C. and Lehnert, W. 1991. A Cognitively Plausible Approach to Understanding Complicated Syntax. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, Anaheim, CA. AAAI Press / MIT Press. 117–124.

(Cardie, 1993a) Cardie, C. 1993a. A Case-Based Approach to Knowledge Acquisition for Domain-Specific Sentence Analysis. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, Washington, DC. AAAI Press / MIT Press. 798–803.

(Cardie, 1993b) Cardie, C. 1993b. Using Decision Trees to Improve Case-Based Learning. In Utgoff, P., editor, *Proceedings of the Tenth International Conference on Machine Learning*, University of Massachusetts, Amherst, MA. Morgan Kaufmann. 25–32.

(Daelemans *et al.*, 1997)
Daelemans, W.; Bosch, A.van den; and Weijters, T. 1997. IGTree: Using Trees for Compression and Classification in Lazy Learning Algorithms. *Artificial Intelligence Review* 11:407–423.

(Daelemans *et al.*, 1996) Daelemans, W.; Zavrel, J.; P., Berck; and S., Gillis 1996. MBT: A Memory-Based Part of Speech Tagger-Generator. In Ejerhed, Eva and Dagan, Ido, , editor, *Proceedings of the Fourth Workshop on Very Large Corpora.* 14–27.

(Domingos, 1997) Domingos, P. 1997. Context-sensitive feature selection for lazy learners. *Artificial Intelligence Review* 11:227–253.

(Fawcett, 1996) Fawcett, T. 1996. *Learning with skewed class distributions — summary of responses.* Machine Learning List: Vol. 8, No. 20.

(Fisher, 1987) Fisher, D. 1987. Knowledge Acquisition Via Incremental Conceptual Clustering. *Machine Learning* 2:139–172.

(Freund and Schapire, 1996) Freund, Y. and Schapire, R.E. 1996. Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on Machine Learning.* Morgan Kaufmann, San Francisco, CA.

(Friedman, 1994) Friedman, J. H. 1994. Flexible metric nearest neighbor classification. Unpublished manuscript available at playfair.stanford.edu via /pub/friedman/README.

(Hastie and Tibshirani, 1994) Hastie, T. J. and Tibshirani, R. J. 1994. Discriminant adaptive nearest neighbor classification. Unpublished manuscript available at playfair.stanford.edu as /pub/hastie/dann.ps.Z.

(Howe and Cardie, 1997) Howe, N. and Cardie, C. 1997. Examining Locally Varying Weights for Nearest Neighbor Algorithms. In *Proceedings of the Second International Conference on Case-Based Reasoning*, Brown Univeristy, Providence, RI. To appear.

(Lehnert, 1990) Lehnert, W. 1990. Symbolic/Subsymbolic Sentence Analysis: Exploiting the Best of Two Worlds. In Barnden, J. and Pollack, J., editors, *Advances in Connectionist and Neural Computation Theory.* Ablex Publishers, Norwood, NJ. 135–164.

(Lewis and Catlett, 1994) Lewis, D. D. and Catlett, J. 1994. Heterogeneous Uncertainty Sampling for Supervised Learning. In *Proceedings of the Eleventh International Conference on Machine Learning*, Rutgers University, New Brunswick, NJ. Morgan Kaufmann. 148–156.

(McCarthy and Lehnert, 1995) McCarthy, Joseph F. and Lehnert, Wendy G. 1995. Using Decision Trees for Coreference Resolution. In Mellish, C., editor, *Proceedings of the Fourteenth International Conference on Artificial Intelligence.* 1050–1055.

(MUC-5, 1994) *Proceedings of the Fifth Message Understanding Conference (MUC-5).* Morgan Kaufmann, San Mateo, CA.

(Pazzani *et al.*, 1994) Pazzani, M.; Merz, C.; Murphy, P.; Ali, K.; Hume, T.; and Brunk, C. 1994. Reducing Misclassification Costs. In *Proceedings of the Eleventh International Conference on Machine Learning*, Rutgers University, New Brunswick, NJ. Morgan Kaufmann. 217–225.

(Quinlan, 1992) Quinlan, J. R. 1992. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo, CA.

(Soderland and Lehnert, 1994) Soderland, S. and Lehnert, W. 1994. Corpus-Driven Knowledge Acquisition for Discourse Analysis. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, Seattle, WA. AAAI Press / MIT Press. 827–832.

(Stanfill and Waltz, 1986) Stanfill, C. and Waltz, D. 1986. Toward Memory-based Reasoning. *Communications of the ACM* 29:1213–1228.

(Wettschereck *et al.*, 1997) Wettschereck, D.; Aha, D. W.; and Mohri, T. 1997. A review and empirical evaluation of feature-weighting methods for a class of lazy learning algorithms. *Artificial Intelligence Review* 11:273–314.

(Wettschereck, 1994) Wettschereck, D. 1994. *A Study of Distance-Based Machine Learning Algorithms.* Ph.D. Dissertation, Dept. of Computer Science, Oregon State University, Corvallis, Oregon.