Two Case Studies in Cost-Sensitive Concept Acquisition*

Ming Tan and Jeffrey C. Schlimmer
School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213
tan@cs.cmu.edu, schlimmer@cs.cmu.edu

Abstract

This paper explores the problem of learning from examples when feature measurement costs are significant. It then extends two effective and familiar learning methods, ID3 and IBL, to address this problem. The extensions, CS-ID3 and CS-IBL, are described in detail and are tested in a natural robot domain and a synthetic domain. Empirical studies support the hypothesis that the extended methods are indeed sensitive to feature costs: they deal effectively with varying cost distributions and with irrelevant features.

Introduction

Consider a simple mobile robot whose primary goal is to pick-up cups and discard them. At some point the robot must determine which of its sensing procedures allow it to distinguish between different types of cups and other objects. In the general situation, an agent must decide which of its available information resources are actually useful. If sensing procedures could be executed instantaneously, this problem would be greatly simplified (thus this assumption is frequently made); however, sensing procedures may incur significant expense, limiting the ability of the robot to routinely execute large numbers of sensing procedures.

As the robot determines the relationships between sensing procedures and object types, it is learning from examples (LFE), where objects in its environment map to examples, sensing procedures map to features of examples, and appropriate ways to pick up objects map to classes of examples. Like sensing procedures, measuring features of examples may also be expensive. From this point of view, researchers have typically studied a degenerate case of LFE where feature expense is assumed to be negligible, and thus all (or nearly all) features are evaluated for each example. In general, individual features incur different measurement costs, and thus *cost-sensitive* learning methods must limit the overall expense of evaluating features during construction and use of class descriptions. (For convenience, we refer to feature measurement cost as feature cost.) This may involve evaluating many inexpensive features or a few expensive ones as the situation dictates. This paper presents two incremental, cost-sensitive LFE methods, CS-ID3 and CS-IBL, generalizations of ID3 (Quinlan 1986) and IBL (Aha & Kibler 1989), and it empirically evaluates the four methods in the robot's domain and a synthetic domain.

*This research is supported by the National Science Foundation under grant IRI-8740522 and by a grant from Digital Equipment Corporation.

Cost-Sensitive Learning Methods

Two notable cost-sensitive learning methods are Nunez's (1988) extension to ID3 and Gennari's (1989) focus-of-attention extension to COBWEB. Nunez's approach focuses on minimizing feature evaluation for test examples (i.e., during classification) but assumes that all features are evaluated for training examples (i.e., during learning). As such, it represents more of an approach to cost-sensitive classification rather than cost-sensitive learning. Gennari's approach, while addressing the more complex task of concept formation, assumes it is sufficient to reduce the number of features evaluated, independent of individual feature costs. Both of the LFE methods we describe here relax this latter assumption, instead assuming that features may vary in cost, and they attempt to reduce the overall cost of evaluating features during both construction and use of class descriptions. After describing the motivating domain in some detail, this section focuses on two effective LFE methods, namely ID3 and IBL, and presents extensions designed to make them sensitive to feature costs during learning and classification.

The Robot Cup Collecting Domain

Consider again the robot whose task is to collect discarded cups. For this task we have experimented with a Heath Hero 2000, a mobile robot with a five degree of freedom arm (torso, arm, elbow, pitch, and roll) and a two finger hand. The Hero robot has a wrist sonar with range of 127 and accuracy of 0.5 inches. To locate objects, the robot uses a ceiling-mounted vision system and a heuristic path-planner.

We have currently defined four sensing procedures for the robot by composing arm movements with intermittent sonar sensings: two forward-facing, rotating

Table 1. ID3's feature selection measure, where F is a feature, V_i is the ith value of F, and $\{C\}$ is the set of classes.

$$I(F) = \left[\sum_{i\in\{C\}} -p(C_i)\log_2 p(C_i)\right] - \left[\sum_{i\in F} p(F=V_i) \sum_{j\in\{C\}} -p(C_j|F=V_i)\log_2 p(C_j|F=V_i)\right]$$

sweeps (bottom-up and left-to-right) – `V-sweep` and `H-sweep`, and two translating scans (forward-facing and downward-facing) – `H-scan` and `D-scan`. The costs of the sensing procedures vary from 28 to 60 seconds. Each sensing procedure yields one or more sensory-based features of the environment.[1] For instance, `V-sweep` yields the `height-deg` feature. We have also defined five specific instantiations of more general grasping procedures for the robot: a front-facing, wide-open grasp – `front-wrap`, two overhead, downward, wide-open grasps – `top-grip` and `top-wrap`, and two overhead, edge grasps – `low-pinch` and `high-pinch`.

There are seven types of objects in the robot's world, five to be grasped (32 oz. plastic cups standing or lying down, 16 oz. waxed paper cups, 8 oz. Styrofoam cups, and tennis ball cans) and two to be ignored (rectangular boxes standing or lying down). Appropriate actions for these objects are `high-pinch`, `top-wrap`, `low-pinch`, `top-grip`, `front-wrap`, `ignore`, and `ignore`, respectively.

To map the robot's task into LFE, we utilize additional knowledge about the task. Specifically, sensing procedures yield different values at different places because objects' appearances vary with viewing angle and distance. To reduce this 'noise,' sensing procedures are instantiated at particular distances and orientations with respect to the object measured. Rather than instantiate sensing procedures at all distances and orientations, a domain-dependent procedure identifies a few, preferred distances and orientations (Tan 1990). For the sensing procedures and objects listed above, this yields three preferred, relative distances (6, 13, and 26 inches) and four relative orientations (0, 90, 180, and 270 degrees). Therefore, the possible features of an example are the sensor-based features resulting from applying sensing procedures at each distance-orientation pair. The class of an example is simply its appropriate grasping procedure.

Feature costs arise from the complexity of a sensing procedure and the amount of motion required to get to a preferred distance-orientation pair. Moving costs are roughly proportional to distance: for 12 inches, the robot requires approximately 30 seconds to aim, move, and position itself. However, as the robot confronts its environment, it may encounter obstacles that force additional navigation and increase moving costs.

This domain requires sensitivity to the costs of measuring features. Brute-force methods that apply all sensing procedures as they build class descriptions would take more than 30 minutes to gather data on each potential cup. Cost-sensitive methods illustrate that less than 2 minutes is typically needed. In general, without cost-sensitive learning methods, users are penalized for including additional, potentially useful features and are forced to determine feature relevance, a job perhaps better left to the learning method.

Learning Decision Trees

ID3 is a LFE method that constructs decision trees to represent concepts (Quinlan 1986). Given a training set of examples with their features and classes, it produces a discrimination tree whose nodes indicate useful feature tests and whose leaves assign classes to examples. ID3 uses a divide-and-conquer algorithm, selecting feature tests to divide the training set and recursively building subtrees to describe partitions. To select new features, ID3 applies an information theoretic measure to estimate the correlation between example features and classes. Specifically, ID3 selects the feature that maximizes the equation in Table 1. The process terminates when the training examples have the same class or when there are no more features to test. To classify new examples, ID3 repeatedly tests the feature at the current subtree root and follows the matching branch. When it reaches a leaf, it predicts that the new example's class is the most common.

Cost-Sensitive ID3 First, to make ID3 cost-sensitive, its feature selection measure should be a function of feature costs as well as I. Nunez (1988) uses this approach, making feature selection inversely proportional to cost. Our cost-sensitive ID3 (CS-ID3) follows Nunez's approach and uses the function I^2/C, where C is the cost of evaluating a feature.

Second, unlike ID3 and Nunez's work, CS-ID3 cannot assume that all features are evaluated for all examples, so during learning its termination criteria must consider *empty* examples which have only a class label but no feature values. In this case, if there is a class ambiguity at a leaf, CS-ID3 continues discrimination by evaluating the next least expensive feature. If fur-

[1] These features are noisy and are filtered by sonar-specific parameters (cf. Tan & Schlimmer 1989).

ther discrimination is required, CS-ID3 evaluates the next least expensive feature, and so on. This greedy strategy is biased toward using many, inexpensive features and against using a few, expensive ones.

Third, to facilitate comparison with incremental systems, CS-ID3 incrementally processes its training examples in two aspects: (a) it does not evaluate newly suggested features for previous examples, and (b) it incrementally revises its decision tree based on features currently preferred by the feature selection measure. The first constraint is enforced by evaluating for a new example only the features referenced by the decision tree during the new example's classification. A resulting implication is that CS-ID3 may require significantly more examples to converge than ID3. Also, training examples may not have a value for each tested feature; these partially-valued examples are used to compute the feature selection measure but are otherwise ignored. For the second constraint, ID5 demonstrates a computationally efficient method to revise trees (Utgoff 1989), but given our interest here in environmental costs and for simplicity of implementation, CS-ID3 simply rebuilds trees from scratch.

Detailed Example Returning to the robot's domain, consider how CS-ID3 builds and uses a decision tree to determine how to grasp the seven object types. Assume the robot is 35 inches away from and at a 270° orientation to the objects. The first object is a box lying down. Since CS-ID3 has no prediction, it simply saves this empty example. The second object is a standing box. CS-ID3 builds a degenerate, empty tree, correctly predicts that this object should also be ignored, and saves it in the training set. The third object is a tennis ball can. This time the prediction is incorrect, and to distinguish between objects to be ignored and those grasped with `front-wrap`, CS-ID3 applies the cheapest sensory procedure, `V-sweep`, at the most convenient, preferred distance and orientation, 26 inches and 270°. The resulting value of 17 for the feature `height-deg` is added to this example before it is saved. The fourth object is an 8 oz. cup. The simple tree applies `V-sweep` and returns a value of 6 for `height-deg`; CS-ID3 incorrectly predicts the tennis can's grasping procedure. The values 17 and 6 are sufficient for discrimination, so no other features are evaluated before the fourth object is stored. When the fifth object (16 oz. cup) is processed, CS-ID3's tree applies `V-sweep` and gets a value of 12. Because the current tree splits `height-deg` at 11.5, CS-ID3 incorrectly predicts the tennis ball can class. CS-ID3 could discriminate between 12 and 6 with an additional binary split, but it prefers to find features that discriminate in a single split. It applies the next cheapest sensory procedure `V-sweep` at the next closest distance 13 inches before storing the fifth object.

As more examples are processed, CS-ID3's feature selection measure becomes more useful, and after 21 examples it converges to the tree depicted in Figure 1. Note that CS-ID3 prefers to reuse features (with their zero subsequent cost). For comparison to CS-IBL (in the following section), after 35 examples, CS-ID3 has made 8 prediction errors, saved all 35 examples, and applied sensory procedures an average of 1.5 times per example.

Learning Instance-Based Descriptions

Like ID3, instance-based learning (IBL) is also an effective LFE method (Aha & Kibler 1989). In its simplest form, instead of constructing an explicit, abstract representation of the training examples, IBL simply saves examples and relies on abstract matching. Given a new example to classify, IBL finds the most similar example in memory and predicts its class for the new example. The similarity measure IBL uses is the negation of the Euclidean distance between two examples, where numeric features are normalized to the range [0,1] and non-numeric features have a difference of 0 if equal in value and 1 otherwise. At its simplest IBL saves all new examples, but it is more effective to save only those which were incorrectly predicted. Other, more sophisticated extensions are possible, and they appear to improve performance in difficult learning situations (Aha & Kibler 1989).

Cost-Sensitive IBL To make IBL cost-sensitive, the classification process it uses to find similar, stored examples must specify which features (and how many) should be evaluated for a new example. Following the spirit of IBL, our approach (CS-IBL) uses stored examples to serve as templates for feature evaluation. Instead of evaluating all features of all stored examples, CS-IBL repeatedly selects one stored, cost-effective example and evaluates one of its features for the new example until the closest example has been found.

First, because all stored examples are equally close to a new, empty example, CS-IBL selects the closest example that: (a) has features that are not yet evaluated for the new example, (b) has common feature values, and (c) uses inexpensive features. Specifically, CS-IBL selects the example that maximizes the ratio of expected match success to cost. The former is estimated by the product of independent feature-value probabilities, and the latter, by summing over feature costs times their decreasing likelihood of being evaluated. Eq. 1 implements this ratio, where P_{ij} is the frequency that feature j has the value it does for stored example i, $\{F'\}$ is the set of features evaluated for the stored example but not for the new example (sorted

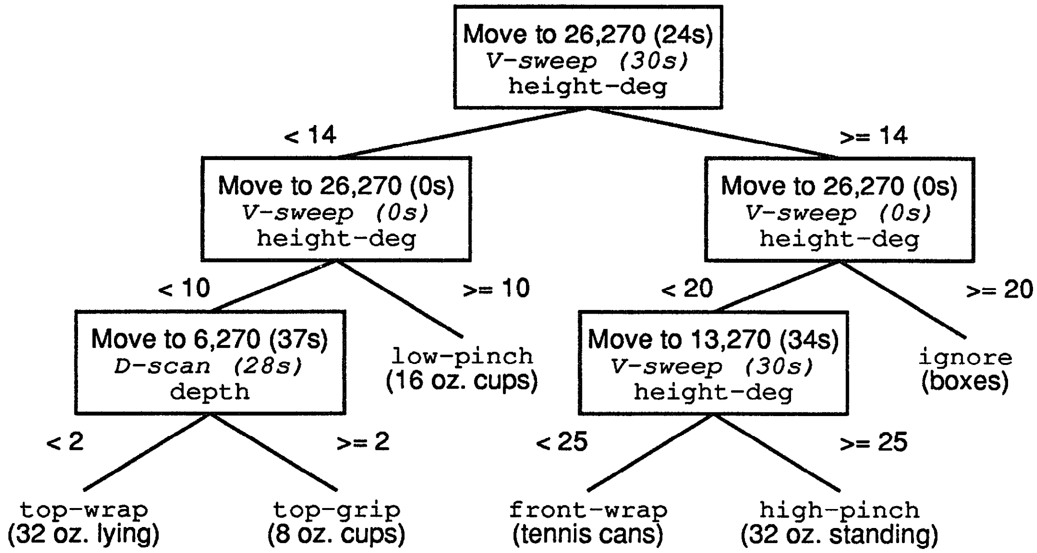

Figure 1. CS-ID3's decision tree for distance = 35 inches, orientation = 270°. Within nodes lines are: the appropriate motion command and its cost in seconds, the sensory procedure to be applied, and the feature to be evaluated. Leaves in the tree correspond to predicted grasping procedures.

in decreasing order of $P_{ij}/C(F_i)$) (cf. Cox 1988), and $C(F_j)$ is the cost of evaluating feature j.

$$\frac{\prod_{j\in\{F'\}} P_{ij}}{\sum_{j\in\{F'\}} C(F_j) \times \prod_{k=1}^{j-1}(1 - P_{ik})} \qquad (1)$$

Second, given an example to guide feature selection, CS-IBL chooses a feature which has a likely value and is inexpensive, or maximizes $P_{ij}/C(F_j)$. CS-IBL repeats this and the above step until the upper bound on the distance from the selected stored example is less than the lower bound for one from any other class. This stopping criteria is conservative and reminiscent of Gennari's (1989) criteria for focus-of-attention in COBWEB.

A final modification ensures that CS-IBL does not end up storing a set of empty or insufficiently described examples. After a prediction error, CS-IBL evaluates one or more additional features for the new example if it appears identical to the stored one. New features to evaluate are first drawn from the closest stored example and then on a cost basis until the two examples are sufficiently different.

Detailed Example Returning again to the robot's domain, CS-IBL initially stores examples and evaluates features in a manner similar to CS-ID3, but later processing reveals three primary differences. First, unlike CS-ID3 which saves all objects, CS-IBL does not save those whose class was correctly predicted. Second, CS-IBL is lazier with respect to expanding the set of evaluated features than CS-ID3. Whereas CS-ID3 evaluates more features when a single binary split is insufficient for discrimination, CS-IBL waits until two examples from different classes cannot be discriminated at all. Third, if CS-IBL is mislead by the cost-driven heuristics to evaluate an irrelevant feature value, the mistake is subsequently propagated, and whenever a stored example with that feature is selected, the irrelevant feature will be re-evaluated for new examples. Though not as bad as evaluating all irrelevant features (as IBL does), it is not as good as evaluating none (as CS-ID3 does). CS-IBL converges after processing 35 examples and has made 11 errors, saved 12 examples, and applied sensory procedures an average of 1.7 times per example. Compared to CS-ID3, this represents slower convergence, more errors, fewer saved examples, and comparable numbers of features evaluated. At an abstract level, one difference between the two methods is that CS-IBL avoids committing to a particular test ordering, and we suspect that this may make it easier to tolerate feature noise and dynamic changes in feature costs.

Empirical Evidence

One might expect that LFE methods which evaluate all example features are more expensive and incur fewer errors prior to convergence than cost-sensitive methods. The empirical results of this section bear this out. This leaves two open questions: how cost efficient are the methods, and how many errors do they incur prior to convergence given different feature cost distributions and numbers of irrelevant features? To provide

Table 2. Performance of the four learning methods in the robot's domain.

	0 Irrelevant Features		12 Irrelevant Features	
	Average Cost (sec)	Total Errors	Average Cost (sec)	Total Errors
ID3	2111.0	5.5	3118.5	5.5
CS-ID3	101.8	9.3	106.1	9.7
IBL	2111.0	6.0	3118.5	6.0
CS-IBL	100.5	11.6	109.1	15.5

initial answers to these questions, this section compares ID3 and three incremental methods CS-ID3, IBL, and CS-IBL in the robot's domain and in a synthetic one. For these studies, ID3 is applied as a brute-force incremental method, rebuilding the tree from scratch after predicting the class of each new example.

Revisiting The Robot's Domain

Applying the four methods to the robot's domain generally indicates that the cost-sensitive methods are more efficient but may incur more errors prior to convergence. Table 2 summarizes two dependent measures prior to convergence given 24 relevant features plus 0 or 12 irrelevant features: (a) the average cost of feature evaluation per example during classification and learning, and (b) the total errors incurred. Given an initial distance of 35 inches, data are an average over five object orders and the four preferred orientations. Irrelevant features were designed to simulate out-of-range sonar readings and had constant values; these features were assigned random costs consistent with the variance of other feature costs. Note that the cost-sensitive methods are highly cost-efficient compared to ID3's and IBL's strategy of evaluating all features for each example, representing an order of magnitude savings. In terms of errors, ID3 and IBL outperform their cost-sensitive derivatives by approximately two to one. Further note that the cost-sensitive methods appear to scale well given irrelevant features; adding 50% more features results in at most a 9% increase in average cost and at most a 33% increase in errors. This latter observation is somewhat surprising and is investigated further in the next subsection.

Using A Synthetic Domain

The robot domain reflects realistic properties of LFE, but it can be difficult to accurately assess those properties. Synthetic domains, conversely, afford precise control over experimental conditions. The experiments in this subsection use a simple Boolean concept, $(A \land B) \lor (C \land D)$, to study the effects of differing costs and numbers of irrelevant features. Given four

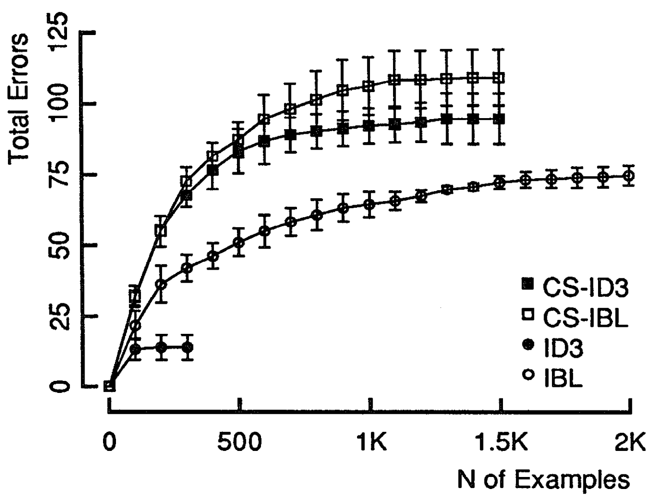

Figure 2. Total errors prior to convergence given four irrelevant features and moderately uneven feature costs.

irrelevant features and moderately uneven costs (Condition 3 below), Figure 2 depicts total errors prior to convergence for each of the four methods ID3, CS-ID3, IBL, and CS-IBL.[2] ID3 yields the fewest errors; IBL comes in second (also the slowest to converge) with CS-ID3 and CS-IBL bringing up the rear.

For the same conditions, Figure 3 depicts the number of features measured by each of the methods as training progresses. The cost-sensitive methods measure considerably fewer features than ID3 and IBL. As both cost-sensitive methods search the space of feature sets, CS-ID3 settles to evaluating a near optimum number of features, but CS-IBL does not. We suspect that this is an artifact of CS-IBL's conservative stopping criteria.

To be efficient, cost-sensitive methods should be sensitive to differential feature costs. Specifically, they should use less expensive features when available. Standard deviation is one quantitative measure of feature cost skew; identical costs have a zero standard deviation, and uneven costs have a standard deviation

[2]Results are averaged over five executions; bars denote standard deviation.

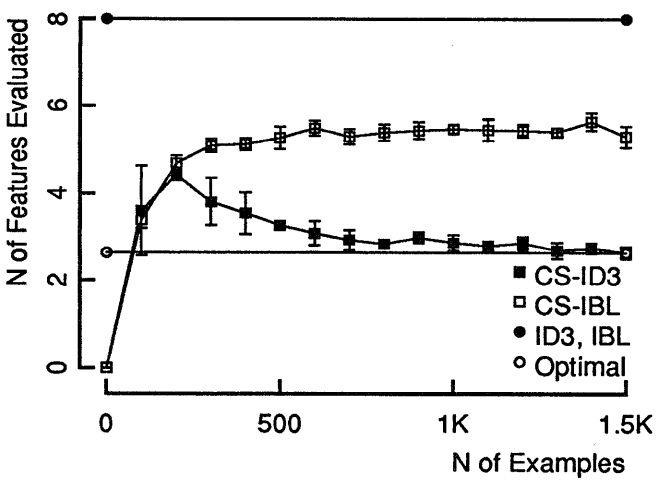

Figure 3. Features evaluated per example prior to convergence given four irrelevant features and moderately uneven feature costs.

much greater than one (when some features are much less expensive than others). Given this metric, we can vary the relative costs of features and observe the resulting cost-sensitive behavior of the four methods. Using the simple Boolean concept above, we divide 40 cost units among groups of 4 features to yield 4 conditions: (1) each feature costs 10, (2) 1 feature costs 1, and 3 features cost 13, (3) 2 features cost 1, and 2 features cost 19,[3] and (4) 3 features cost 1, and 1 feature costs 37. When irrelevant features are included, they are assigned the same costs as relevant features. Given four irrelevant features, Figure 4 depicts the average feature costs per example, and Figure 5, total errors prior to convergence as a function of different relative feature costs. In terms of average cost, both cost-sensitive methods exhibit close to optimal asymptotic performance. They are also considerably less than ID3 and IBL. In terms of errors, the methods separate naturally into three groups, from best to worst, ID3, IBL, and the cost-sensitive methods. This latter, poor performance may arise because the cost-sensitive methods must search for an effective set of features to evaluate.

Cost-sensitive methods should also be sensitive to the number of features as focus-of-attention methods are (cf. Gennari 1989). Using the simple Boolean concept, we added 0, 2, 4, and 8 irrelevant features that have random binary values. Given moderately uneven costs (Condition 3), Figure 6 depicts average feature costs per example, and Figure 7, total errors prior to convergence as a function of the number of irrelevant features. In terms of costs, the cost-sensitive methods

[3]For simplicity of analysis, in this condition features A and B cost 1.

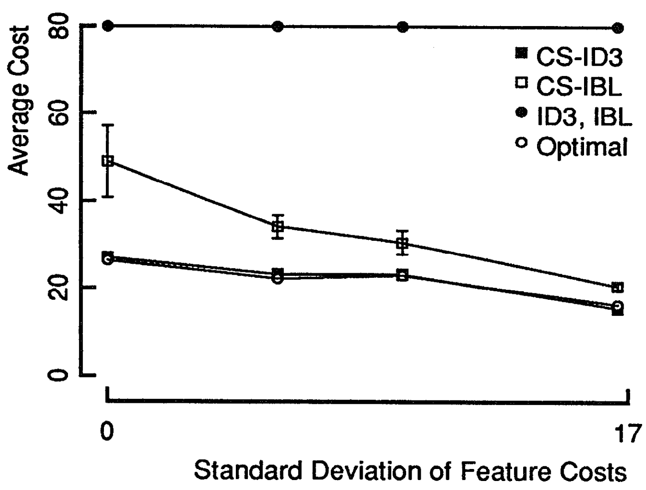

Figure 4. Average feature cost per example prior to convergence given four irrelevant features.

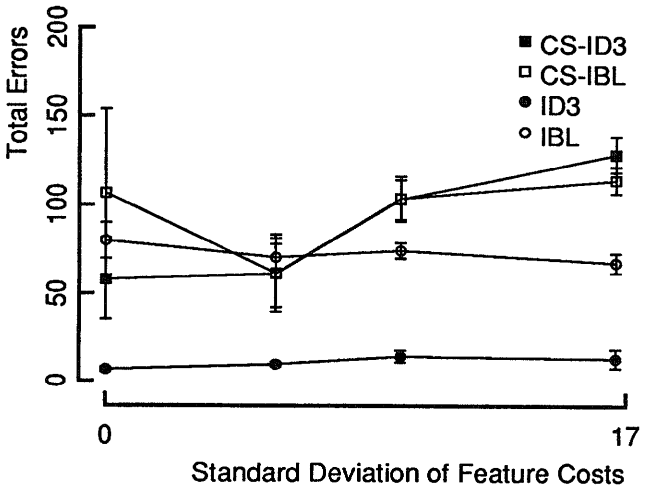

Figure 5. Total errors prior to convergence given four irrelevant features.

again perform at a near optimal level. (CS-ID3 appears slightly below due to its early behavior.) In terms of errors, the methods appear to fall into three groups, from best to worst: ID3, CS-ID3, and the instance-based methods. Both of the instance-based methods incur a sharply increasing number of errors as irrelevant features increase, something that may be remedied by more sophisticated version of IBL (cf. Aha Kibler 1989). Unlike the lower performance of CS-ID3 compared to ID3, CS-IBL and IBL appear equal.

Summary

This paper addresses the general problem of learning from examples when features have non-trivial costs. Though this work utilizes inductive methods, comple-

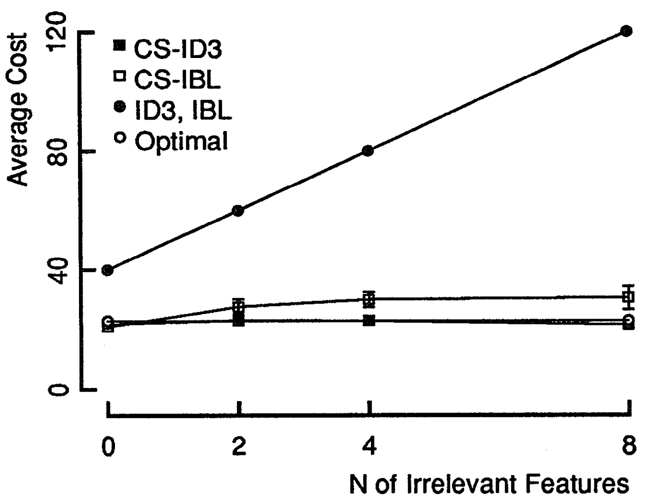

Figure 6. Average feature costs per example prior to convergence given moderately uneven costs.

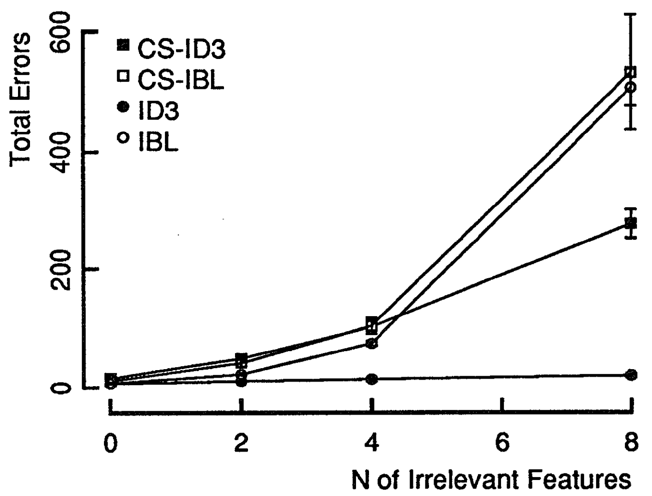

Figure 7. Total errors prior to convergence given moderately uneven costs.

mentary research has investigated similar issues using analytic, or explanation-based learning methods. For instance, Keller (1987) describes a system that trades off the operationality of a concept description for predictive accuracy: typically expensive, fine-grained tests are pruned for the sake of overall improvement. Like Keller's system, CS-ID3 and CS-IBL also attempt to make concept descriptions more operational by minimizing feature measurement costs, but they do not trade off cost for accuracy.

Despite encouraging empirical evidence supporting the hypothesis that CS-ID3 and CS-IBL are sensitive to costs, there are still several open questions: how relevant is CS-IBL's classification flexibility (as compared to CS-ID3), how can cost-sensitive methods reason about parallel feature evaluation, and can cost-sensitive methods tolerate noise? Notwithstanding these, modifying methods to deal with feature costs appears feasible, and we suspect necessary, in future machine learning research.

Acknowledgements We would like to thank Tom Mitchell and Long-Ji Lin for the robotics hardware used in this research, and David Aha, Rich Caruana, Klaus Gross, and Tom Mitchell for their comments on a draft of this paper. Thanks also to the 'gripe' group for providing a consistent and reliable computing environment.

References

Aha, D. W., & Kibler, D. 1989. Noise-Tolerant Instance-Based Learning Algorithms. In Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, 794–799. Detroit, MI: Morgan Kaufmann.

Cox, L. A. 1988. Designing Interactive Expert Classification Systems That Acquire Expensive Information 'Optimally.' In Proceedings of the European Knowledge Acquisition Workshop for Knowledge-Based Systems. Bonn, Germany.

Gennari, J. H. 1989. Focused Concept Formation. In Proceedings of the Sixth International Workshop on Machine Learning, 379–382. Cornell, NY: Morgan Kaufmann.

Keller, R. M. 1987. Defining Operationality for Explanation-Based Learning. In Proceedings of the Sixth National Conference on Artificial Intelligence, 482–487. Seattle, WA: Morgan Kaufmann.

Nunez, M. 1988. Economic Induction: A Case Study. In Proceedings of the Third European Working Session on Learning, 139–145. Glasgow, Scotland: Pitman.

Quinlan, J. R. 1986. Induction of Decision Trees. *Machine Learning* 1(1):81–106.

Tan, M. 1990. CSL: A Cost-Sensitive Learning System for Sensing and Grasping Objects. In Proceedings of the 1990 IEEE International Conference on Robotics and Automation. Cincinnati, OH.

Tan, M., and Schlimmer, J. C. 1989. Cost-Sensitive Concept Learning of Sensor Use in Approach and Recognition. In Proceedings of the Sixth International Workshop on Machine Learning, 392–395. Cornell, NY: Morgan Kaufmann.

Utgoff, P. E. 1989. Incremental Induction of Decision Trees. *Machine Learning* 4(2):161–186.