Constrained Cascade Generalization of Decision Trees

Huimin Zhao, *Member*, *IEEE*, and Sudha Ram, *Member*, *IEEE*

Abstract—While *decision tree* techniques have been widely used in *classification* applications, a shortcoming of many decision tree inducers is that they do not learn intermediate concepts, i.e., at each node, only one of the original features is involved in the branching decision. Combining other classification methods, which learn intermediate concepts, with decision tree inducers can produce more flexible decision boundaries that separate different classes, potentially improving classification accuracy. We propose a generic algorithm for *cascade generalization* of decision tree inducers with the *maximum cascading depth* as a parameter to constrain the degree of cascading. Cascading methods proposed in the past, i.e., *loose coupling* and *tight coupling*, are strictly special cases of this new algorithm. We have empirically evaluated the proposed algorithm using logistic regression and C4.5 as base inducers on 32 UCI data sets and found that neither loose coupling nor tight coupling is always the best cascading strategy and that the maximum cascading depth in the proposed algorithm can be tuned for better classification accuracy. We have also empirically compared the proposed algorithm and *ensemble methods* such as *bagging* and *boosting* and found that the proposed algorithm performs marginally better than bagging and boosting on the average.

Index Terms—Machine learning, data mining, classification, decision tree, cascade generalization.

1 Introduction

Classification is a type of *prediction problem* where the *dependent variable* (also referred to as *class*) that needs to be predicted based on several *independent variables* (also referred to as *features* and *attributes*) is discrete. A related problem is *regression* where the dependent variable is continuous. A *learning technique* constructs a hypothetical model, which is a mapping from the independent variables to the dependent variable, by investigating a given set of successfully *solved cases*, whose outputs on the dependent variable are already known; the model can then be used to predict the outputs of unseen cases on the dependent variable. *Decision tree* techniques follow a "divide and conquer" strategy and produce sequential models which logically combine a sequence of simple tests. They have been popular in classification applications largely because the models they generate closely resemble human reasoning and are easily understood [25].

While d*ecision tree* techniques have been widely used in classification applications, a shortcoming of many decision tree inducers is that they do not learn intermediate concepts, i.e., at each tree node, only one of the original features is involved in the branching decision [3], [5], [11], [12], [14], [16], [19], [20], [25], [29]. Geometrically, the decision boundaries in the feature space are restricted to be orthogonal to the splitting feature's axis. Such *representational bias* limits the ability of decision trees to fit the training data. Combining other classification methods, which learn intermediate concepts, with decision tree inducers can produce more flexible decision boundaries that separate different classes, potentially improving classification accuracy. Gama and Brazdil [12] have named such generalization of decision tree inducers as *cascade generalization*.

In this paper, we propose a generic algorithm for cascade generalization of decision tree inducers with the *maximum cascading depth* as a parameter to constrain the degree of cascading. Cascading methods proposed in the past, i.e., *loose coupling* and *tight coupling*, are strictly special cases of this new algorithm. We have empirically evaluated the proposed algorithm using logistic regression and C4.5 as base inducers on 32 data sets for classification problems in the UC Irvine machine learning repository [2]. Our evaluation results show that neither loose coupling nor tight coupling is always the best cascading strategy and that the maximum cascading depth in the proposed algorithm can be tuned for better classification accuracy. We have also empirically compared the proposed algorithm and *ensemble methods* such as *bagging* and *boosting* and found that the proposed algorithm performs marginally better than bagging and boosting on the average.

The paper is organized as follows: In the next section, we briefly review some related classification methods, including decision tree induction, logistic regression, cascade generalization, and ensemble methods. In Section 3, we propose a new algorithm of cascade generalization. We then report on some empirical evaluation using UCI data sets in Section 4. Finally, we conclude the paper and discuss future research directions in Section 5.

- *H. Zhao is with the School of Business Administration, University of Wisconsin–Milwaukee, PO Box 742, Milwaukee, WI 53201. E-mail: hzhao@uwm.edu.*
- *S. Ram is with the Department of Management Information Systems, Eller School of Business and Public Administration, University of Arizona, 1130 E. Helen, Tucson, AZ 85721-0108. E-mail: ram@bpa.arizona.edu.*

Manuscript received 3 Apr. 2003; revised 21 Oct. 2003; accepted 12 Feb. 2004. For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-0026-0403.

2 Background

Our proposed algorithm extends and generalizes previous algorithms for cascade generalization. Our empirical evaluation involves an implementation of the proposed algorithm using logistic regression and C4.5 decision tree inducer as base inducers and comparisons with regard to ensemble methods such as bagging and boosting. Therefore, in the following sections, we review such related classification methods as decision tree inducers, logistic regression, cascade generalization, and ensemble methods, following a general description of the classification problem and classification algorithms. We then discuss the bias-variance decomposition of classification errors and compare the classification methods with regard to this decomposition.

2.1 Classification

A p-class classification problem is described by a pair $< X, Y >$, where $X = X_1 \times X_2 \times \cdots \times X_m$ is an m-dimensional space and $Y = \{1, 2, \ldots, p\}$ is a discrete space. $X_i (i = 1, 2, \ldots, m)$ is called an *individual feature* (or *attribute*, or *independent variable*) *space* and X the *total feature space*. In the rest of the paper, we use the term *feature space* to refer to the total feature space X when there is no danger of ambiguity. Y is called the *class* (or *dependent variable*) *space*. A *solved case* (also referred to as *instance* or *example*) is a pair $< x, y >$, where $x =< x_1, x_2, \ldots, x_m > \in X$ and $y \in Y$. A *sample* is a set of n solved cases

$$S = \{< x(1), y(1) >, < x(2), y(2) >, \ldots, < x(n), y(n) >\}.$$

A *classification algorithm* (or *inducer*) takes a training sample as input and outputs a *classifier*, which is a mapping $f : X \to Y$. Let $\mathbf{x} = \{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x}_m\}$ and $\mathbf{y}$ denote variables corresponding to the feature vector and class membership of a case. The performance of a classifier f can be measured by *accuracy*, the probability of making a correct prediction when given a case $< \mathbf{x}, \mathbf{y} >$, denoted $accuracy(f) = P(f(\mathbf{x}) = \mathbf{y})$. A related measure, *error rate*, is defined as $error(f) = 1 - accuracy(f)$. Accuracy derived based on the training sample is called *apparent accuracy* and is usually overly optimistic and not reliable. Instead, an independent testing sample should be used to estimate the true accuracy. There are also more reliable techniques such as *cross-validation* and *bootstrap* for accuracy estimation [17]. A problem to overcome in designing classification algorithms is *overfitting*, i.e., the learned classifier has high apparent accuracy but low estimated true accuracy. Note that accuracy (or error rate) is a special case of a more general performance measure, *expected misclassification cost*, when the costs of different types of classification errors are considered equal in the context [26]. In our empirical evaluation, we use accuracy as the sole performance measure because it is impossible to allocate a uniform cost matrix across the data sets drawn from different application domains.

Theoretically, an optimal classifier (called *Bayes optimal classifier*) that minimizes error rate exists and is equivalent to the following mapping:

$$\begin{aligned} &f^*(x) = i (i = 1, 2, \ldots, p), \\ &\textbf{if } \forall j = 1, 2, \ldots, p; j \neq i \\ &\quad (P(\mathbf{y} = i | \mathbf{x} = x) > P(\mathbf{y} = j | \mathbf{x} = x)). \end{aligned} \tag{1}$$

Applying the Bayes rule, $P(B|A) = \frac{P(A|B)P(B)}{P(A)}$, (1) is equivalent to:

$$\begin{aligned} &f^*(x) = i (i = 1, 2, \ldots, p), \\ &\textbf{if } \forall j = 1, 2, \ldots, p; j \neq i \\ &\quad (P(\mathbf{x} = x | \mathbf{y} = i) P(\mathbf{y} = i) > P(\mathbf{x} = x | \mathbf{y} = j) P(\mathbf{y} = j)). \end{aligned} \tag{2}$$

Solving the problem requires determination of prior probabilities $P(\mathbf{y})$ and conditional probabilities $P(\mathbf{x}|\mathbf{y})$. However, in practical classification applications, it is rarely possible to directly estimate $P(\mathbf{x}|\mathbf{y})$, as enumerating the points in X requires an enormous number of sample cases, especially when some of the features are continuous. All practical classification methods can be seen as trying to estimate $P(\mathbf{x}|\mathbf{y})$ with various simplifying assumptions and heuristic search strategies. The assumption with regard to the structure of f made by a classification method is called the *representational bias* of the method.

2.2 Decision Tree Inducers

Most decision tree inducers assume that the prediction decision can be made via a sequence of small tests (or decisions), each of which usually involves a single feature $\mathbf{x}_i$. In the learned classifiers, the decision boundaries that separate different classes in an m-dimensional feature space are $(m-1)$-dimensional hyperplanes that are geometrically orthogonal to the axes of the testing features. Most decision tree inducers also follow a "divide and conquer" heuristic search strategy and can be described in the following generic algorithm (Algorithm 1):

Algorithm 1. (Build Decision Tree)

$Build\_Decision\_Tree(N, < X, Y >, S)$

N: A node in the decision tree to be learned. N is the root node when the procedure is initially invoked. An intermediate node may have an indefinite number q of children, which are denoted $N.Child[i] (i = 1, 2, \ldots, q)$.

X: The feature space of the classification problem. $X = X_1 \times X_2 \times \cdots \times X_m$.

Y: The class space of the classification problem. $Y = \{1, 2, \ldots, p\}$.

S: A sample, i.e., a set of n solved cases, each of which is a pair $< x, y >$, where $x =< x_1, x_2, \ldots, x_m > \in X$ and $y \in Y$.

1 Select a splitting test based on a goodness measure, by which S is split into q subsets $S_i (i = 1, 2, \ldots, q)$.

2 If $q = 1$,

2.1 Mark N as a leaf node, with the majority class in S as the predicted class.

3 Else

3.1 For $i = 1$ To q,

3.1.1 Generate the ith child of N, $N.Child[i]$.

3.1.2 $Build\_Decision\_Tree(N.Child[i], < X, Y >, S_i)$.

Different decision tree inducers mainly differ in the goodness measure used to select the splitting feature (line 1). ID3 selects the feature that results in the biggest *information gain* [21], [22]. C4.5, a successor of ID3, replaces information gain with a *gain ratio* to compensate ID3's bias toward highly-branching features [23]. Many other measures, such as G-statistic and Gini index, have been adopted in other decision tree inducers [19]. The tree building procedure is usually followed by a *pruning* phase, in which some selected subtrees are replaced by single leaves (*subtree replacement*) or "raised" to replace their parents (*subtree raising*), to reduce the chance of overfitting [19], [25], [27]. Pruning will certainly reduce the apparent accuracy on the training sample, but may increase the expected accuracy during prediction.

2.3 Logistic Regression

Logistic regression is a widely-used statistical method for classification. A major difference between logistic regression and *linear regression* is that the dependent variable is discrete in logistic regression and continuous in linear regression. Logistic regression assumes that the *logits*, logarithm of the *odds ratios*, are linear with regard to the features [15], [25]. In a binary classification problem (i.e., $p = 2$), the logits are the following linear functions:

$$\begin{aligned} g_i(x) &= \ln \frac{P(\mathbf{y} = i|\mathbf{x} = x)}{1 - P(\mathbf{y} = i|\mathbf{x} = x)} \\ &= \sum_{j=1}^{m} \beta_{ij} x_j + \beta_{i0} \qquad (\textbf{for } i = 1, 2). \end{aligned} \tag{3}$$

Note that $\beta_{1j} = -\beta_{2j} (j = 1, 2, \ldots, m)$. The decision boundary that separates the classes in the feature space X is linear. The coefficients, $\beta_{ij} (i = 1, 2; j = 1, 2, \ldots, m)$, can be obtained using an *iterative weighted least squares procedure*. Assuming that the conditional distribution of $\mathbf{x}$, given the class membership $\mathbf{y} = y$, is multivariate normal with a covariance matrix that is independent of y, the coefficients can also be estimated analytically using the *linear discriminant analysis* method. Because the assumptions made by linear discriminant analysis do not usually hold, it should be used instead of logistic regression only when the resource is limited and only in preliminary analysis [15]. Logistic regression and linear discriminant analysis both can be easily extended to deal with multiple-class classification problems (i.e., $p > 2$) via multiple pairwise comparisons.

2.4 Ensemble Classification Methods

The fact that a single classifier provides just one estimate of the Bayes optimal classifier (2), no matter how accurate it is, has led researchers to explore methods to obtain better estimates by combining multiple classifiers [6]. One type of these multiple classifier methods is called *ensemble methods* (also called *voting methods*) [1], [6], [7], which make the final decisions based on (*weighted* or *unweighted*) *voting* of a set (called *ensemble* or *committee*) of classifiers. Ensemble methods differ in the way base classifiers are induced and the voting mechanism. Two of the most popular ensemble methods are *bagging* and *boosting*. In bagging (**B**ootstrap **Agg**rega**ting**) [4], T base classifiers $f_i (i = 1, 2, \ldots, T)$ are induced independently using different training samples $S_i (i = 1, 2, \ldots, T)$ and are given equal weights in the voting. The samples are generated based on an original training sample S using the *bootstrap* technique [17]; each time, $|S|$ cases are *randomly sampled with replacement* from S. In boosting, base classifiers are learned sequentially; each new classifier pays more attention to cases misclassified by previous classifiers. In AdaBoost [10], a popular boosting method, each case correctly classified by the ith classifier f_i is down-weighted by a factor of $\frac{error(f_i)}{1-error(f_i)}$ before the induction of f_{i+1}; each classifier f_i is assigned a weight $-\log \frac{error(f_i)}{1-error(f_i)}$ in the voting.

2.5 Cascade Generalization

Cascade generalization is another method for combining classifiers [12]. While several authors have proposed similar ideas independently, the term *cascade generalization* is due to Gama and Brazdil [12]. While bagging and boosting combine classifiers generated by the same inducer, cascade generalization involves multiple inducers; the output of one inducer is used to construct new features, which will be used in addition to the original features by the next inducer. Cascade generalization has usually been used to combine a decision tree inducer with other classification methods.

As we have mentioned earlier, most decision tree inducers have a representational bias; the trees they produce are usually *univariate*; the splitting decision at each internal node is based on a single *feature* [3], [5], [11], [12], [14], [16], [19], [20], [25], [29]. The decision boundaries in the feature space are restricted to be geometrically orthogonal to the axes of the splitting features. In a two-dimensional feature space, the decision regions are rectangles, whose sides are parallel to one of the axes; for a linearly separable data set, they must use numerous axis-parallel line segments to approximate the discriminant line. Recently, several methods have been proposed to cascade other classification techniques, often linear model inducers, with decision tree inducers. The generalized decision trees generated by these methods have been named *multivariate decision trees* [5], *oblique decision trees* [14], [20], *discriminant trees* [11], and *linear discriminant trees* [16], [29]. The splitting decision at an intermediate node is based on a *multivariate* test, which is often a linear combination of the original features, rather than a single feature. The *multivariate* tests are based on classifiers induced by the other inducers cascaded with the decision tree inducer. In a special case, where the splitting decision is base on a linear combination of at most two features, the generalized decision trees are called *bivariate decision trees* [3]. The decision boundaries in an m-dimensional feature space are $(m-1)$-dimensional hyperplanes that are not restricted to be *axis-orthogonal* and can be *oblique*. The representational bias of univariate decision trees is relaxed; training samples can be better fitted (i.e., apparent accuracy tends to increase).

2.6 Bias-Variance Decomposition of Classification Errors

The bias plus variance decomposition [13] is a powerful tool for analyzing supervised learning scenarios that have quadratic loss functions. It has been adapted to deal with zero-one loss functions (i.e., classification error), which is more applicable to classification problems [8], [18]. Given a fixed *target population* of cases, the expected classification error generated by an inducer can be decomposed into three

nonnegative terms: $bias^2$, *variance*, and *noise*. The $bias^2$ term measures the squared difference between the inducer's average guess and the target population's average output. The *variance* term measures the variability of the inducer's average guess across training samples given to the inducer. The *noise* term measures the variance of the target population per se and is independent of the inducer.

The bias-variance decomposition can be used to explain how multiple classifier methods affect classification performance. Empirical studies [1], [4], [6], [7] have shown that: 1) bagging sometimes improves the performance of the base classifiers $f_i(i = 1, 2, \ldots, T)$ and seldom degrades their performance and 2) bagging is more effective with *unstable* (i.e., high variance) inducers such as most decision tree inducers, whose output (i.e., classifiers $f_i(i = 1, 2, \ldots, T)$) fluctuates significantly given small changes in the training sample, than with stable ones such as logistic regression. It has been shown both analytically and empirically that the performance gain of bagging is due to its reduction of variance and, therefore, bagging is more productive on unstable methods [1], [4], [6], [7].

Empirical results [1], [6], [7], [10], [26] regarding boosting include:

1. Boosting sometimes is significantly superior to bagging, but can also go wrong; it sometimes even degrades the performance of the underlying inducer.
2. Boosting is unproductive or even counterproductive with *strong* classifiers.
3. Boosting is sensitive to *noise* and does not work well in highly noisy problems.

It has been shown both analytically and empirically that boosting can reduce both bias and variance and, therefore, generate larger performance gain than bagging [1], [7], [10]. However, boosting can also increase variance when base classifiers are strong, reducing or even overwhelming its performance gain [1]. It also places more weight on "hard" training cases (i.e., cases misclassified by earlier classifiers), which are likely to be noisy cases in noisy problems, artificially increasing the noise.

Empirical studies have shown that multivariate decision trees generated by cascading other inducers with a decision tree inducer frequently outperform univariate decision trees generated by the underlying decision tree inducer [3], [5], [11], [12], [14], [16], [20], [29]. Unlike voting methods, which mainly reduce variance, cascade generalization has been shown to reduce bias [12].

3 Constrained Cascade Generalization

The cascade generalization methods proposed in the past can be categorized into two major types, *loose coupling* and *tight coupling* [12]. In loose coupling, classification methods such as *linear discriminant analysis* and *logistic regression* are used first to learn T initial classifiers $f_i(i = 1, 2, \ldots, T)$. These classifiers are then used to construct m' additional features $\mathrm{x}' = \{\mathrm{x}'_1, \mathrm{x}'_2, \ldots, \mathrm{x}'_{m'}\}$, which are combined with the original features $\mathrm{x} = \{\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_m\}$ to induce a decision tree. In tight coupling, such cascading is localized to each node in the decision tree; additional features are constructed based on the training cases falling into each node. These *constructed features* are functions of the original features and represent intermediate concepts, which relax the representational bias of the base decision tree inducer and allow the decision boundaries to be oblique (rather than axis-orthogonal) hyperplanes in the original feature space X.

Cascade generalization reduces bias and increases the *complexity fit* (or *flexibility*) of the classifiers learned by decision tree inducers so that training data can be fitted better and apparent error rate is reduced. Therefore, tight coupling is more flexible than loose coupling, which, in turn, is more flexible than the underlying univariate decision tree inducer in fitting the training cases. However, no classification method can get away with the trade off between complexity and *generalizability* (i.e., prediction ability on unseen cases) [25]. Generally speaking, the more complex a classifier is, the better it can fit training data, but, at the same time, the easier it is to *overfit* training data [25]. A model that fits training data well may not predict unseen data well. This is especially the case with cascade generalization. As a generalized decision tree grows, the local models at each node are learned based on fewer and fewer training samples. In addition, the training samples get more and more unbalanced toward the *majority class* of the current branch. The cascaded method is more and more prone to learning spurious intermediate models, which overfit the local training cases. One of the major tasks of classification methods that can produce variable complexity fits is to find the appropriate complexity fit [25]. We posit that cascade generalization is not an exception and, therefore, the appropriate complexity fit (i.e., degree of cascading) needs to be determined empirically for a given application.

3.1 A Generic Algorithm for Cascade Generalization

We propose a generic algorithm for cascade generalization, which uses the maximum cascading depth as a parameter to simulate the trade off between complexity and generalizability. The parameter *constrains* cascading to a given extent. An appropriate maximum cascading depth for a particular problem can be found using a performance estimation method such as bootstrap and cross-validation. Our pseudocode for the generic algorithm is presented as a *recursive procedure* called *Build_Generalized_Tree* (Algorithm 2):

Algorithm 2. (Build Generalized Decision Tree)

$Build\_Generalized\_Tree(N, < X, Y >, S, d, d_{cascade}, \Psi, \Gamma)$

$N, < X, Y >, S$: See description under Algorithm 1.

d: The depth of N in the decision tree. $d = 1$ when the procedure is initially invoked.

$d_{cascade}$: The maximum cascading depth. $d_{cascade}$ is a parameter of the algorithm.

Ψ: A base decision tree inducer, e.g, C4.5.

Γ: A set of disciminant function inducers, e.g., {logistic regression}, which takes $< X, Y >$ and S as input and returns a set of constructed features based on the discriminant functions to be learned.

1 $X' := \Phi$.

2 If $d \leq d_{cascade}$,

2.1 $X' := \Gamma(< X, Y >, S)$.

2.2 $X := X \cup X'$.

2.3 $S := S$ extended with constructed features in X'.
3 Select a splitting test using the goodness measure of Ψ, by which S is split into q subsets $S_i (i = 1, 2, \ldots, q)$.
4 If $q = 1$,
4.1 Mark N as a leaf node, with the majority class in S as the predicted class.
5 Else
5.1 For $i = 1$ TO q,
5.1.1 Generate the ith child of N, $N.Child[i]$.
5.1.2 *Build_Generalized_Tree* $(N.Child[i], < \mathrm{X} - X', Y >, \mathrm{S}_i, d + 1, d_{cascade}, \Psi, \Gamma)$.

The algorithm is generic in the sense that it can be used to cascade any number and any kind of *discriminant function* inducers (i.e., the Γ), such as linear discriminant analysis and logistic regression, with any classification scheme (i.e., the Ψ), such as C4.5 [23], that follows a "divide and conquer" strategy and builds *sequential decision models*. The selection of the splitting test at each intermediate node is based on the goodness measure (e.g., *information gain* and *gain ratio*) of the underlying decision tree inducer. Loose coupling and tight coupling proposed in the past [12] are special cases of the algorithm when $d_{cascade} = 1$ and $d_{cascade} = \text{the height of the tree}$, respectively. The procedure *Build_Generalized_Tree* constructs a decision tree; our algorithm is open to any tree *pruning* strategy.

The time complexity of the algorithm can be estimated in terms of the time complexity of the base classification methods. Assume that the training time of the base decision tree method is a function of the number of training instances, n, and the number of features, m, and is denoted $f_\Psi(n, m)$. Assume that the training time of Γ is a function of n and m and is denoted $f_\Gamma(n, m)$. Suppose there are q_d nodes, each with $n_i (i = 1, 2, \ldots, q_d; \sum_{i=1}^{q_d} n_i = n)$ training examples, on some level $d \le d_{cascade}$ of the tree. The total time needed to train Γ for the q_d nodes on level d of the tree is:

$$f'_{\Gamma,d}(n, m) = \sum_{i=1}^{q_d} f_\Gamma(n_i, m) \le f_\Gamma(\sum_{i=1}^{q_d} n_i, m) = f_\Gamma(n, m). \quad (4)$$

We reasonably assume that the inequality holds for $f_\Gamma(n, m)$, as the time complexity functions for most, if not all, classification methods are at least linear with regard to n and m. The inequality approaches equality when the q_d nodes are more unbalanced (i.e., the n_i's are more different). The total time attributed to training Γ on all $d_{cascade}$ levels of the tree is:

$$\begin{aligned} f'_\Gamma(n, m) &= \sum_{d=1}^{d_{cascade}} f'_{\Gamma,d}(n, m) \le \sum_{d=1}^{d_{cascade}} f_\Gamma(n, m) \\ &= d_{cascade} f_\Gamma(n, m). \end{aligned} \quad (5)$$

The time needed to train the generalized decision tree increases slightly, compared to training a decision tree without cascading Γ, due to the additional constructed features added to the training data during the training process. However, when the number of original features is much larger than the number of constructed features, $m' = |\Gamma(< X, Y >, S)|$ (i.e., $m >> m'$), as it is usually true, this increase is negligible and the time attributed to decision tree training is:

$$f'_\Psi(n, m + m') = f_\Psi(n, m + m') \approx f_\Psi(n, m). \quad (6)$$

The overall training time is therefore:

$$\begin{aligned} f'_{d_{cascade}}(n, m) &= f'_\Gamma(n, m) + f'_\Psi(n, m + m') \le d_{cascade} f_\Gamma(n, m) \\ &\quad + f_\Psi(n, m + m') \approx d_{cascade} f_\Gamma(n, m) + f_\Psi(n, m). \end{aligned} \quad (7)$$

If the time complexity of the base decision tree method is on a higher degree than that of Γ (i.e., $f_\Psi(n, m) >> f_\Gamma(n, m)$), the overall training time is dominated by decision tree training and increases only slightly:

$$f'_{d_{cascade}}(n, m) \le d_{cascade} f_\Gamma(n, m) + f_\Psi(n, m) \approx f_\Psi(n, m).$$

If the time complexity of Γ is on a higher degree than that of the base decision tree method (i.e., $f_\Gamma(n, m) >> f_\Psi(n, m)$), the overall training time is bounded by $d_{cascade}$ times of the training time of Γ:

$$f'_{d_{cascade}}(n, m) \le d_{cascade} f_\Gamma(n, m) + f_\Psi(n, m) \approx d_{cascade} f_\Gamma(n, m).$$

In any case, the efficiency of the cascade generalization algorithm is considered acceptable given efficient base classifiers, Ψ and Γ.

3.2 Finding the Best Cascading Depth

Using the generalized decision tree building algorithm, a set of k generalized decision trees with different $d_{cascade}$ values can be built, from which the best tree can be identified using a performance estimation method such as bootstrap and cross-validation [17]. A general procedure is described in the following algorithm (Algorithm 3):

Algorithm 3. (BuildGeneralized Decision Trees)
$Build\_Generalized\_Trees(< X, Y >, S, \Psi, \Gamma, \Omega)$

$< X, Y >, S$: See description under Algorithm 1.
Ψ, Γ: See description under Algorithm 2.
Ω: A performance estimation method, e.g., cross-validation.
1 $i := 0$.
2 Loop
2.1 Generate the root node for the ith generalized tree, R_i.
2.2 *Build_Generalized_Tree* $(R_i, < \mathrm{X}, Y >, \mathrm{S}, 1, i, \Psi, \Gamma)$.
2.3 Estimate the performance of R_i using Ω.
2.4 $i := i + 1$.
Until $R_{i-1}.\mathrm{depth} = i - 1$ (i.e., the last tree is fully cascaded with Γ).
3 Find the best tree among the k trees, $R_i (i = 1, 2, \ldots, k)$.

The time complexity of *Build_Generalized_Trees* is:

$$\begin{aligned} f_{trees}(n, m) &= \sum_{i=0}^{k-1} f'_i(n, m) \le \sum_{i=0}^{k-1} (i \cdot f_\Gamma(n, m) + f_\Psi(n, m)) \\ &= \frac{(k-1)k}{2} f_\Gamma(n, m) + k f_\Psi(n, m). \end{aligned} \quad (8)$$

TABLE 1
Characteristics of 32 UCI Data Sets

No.	Name	#Classes	#Instances	#Features		
				Nominal	Numeric	Total
1	Annealing	6	898	29	9	38
2	Audiology	24	226	69	0	69
3	Automobile	7	205	10	16	26
4	Balance Scale	3	625	0	4	4
5	Wisconsin Breast Cancer	2	699	0	10	10
6	Breast Cancer	2	286	9	0	9
7	Chess (King Rook vs King Pawn)	2	3,196	36	0	36
8	Congressional Voting Records	2	435	16	0	16
9	Credit Card Approval	2	690	9	6	15
10	Statlog Project: German Credit	2	1,000	13	7	20
11	Pima Indians Diabetes	2	768	0	8	8
12	Glass Identification	7	214	0	9	9
13	Heart Disease (Cleveland)	5	303	6	7	13
14	Heart Disease (Hungarian)	5	294	6	7	13
15	Hepatitis	2	155	13	6	19
16	Horse Colic	2	368	15	7	22
17	Image Segmentation	7	2,310	0	19	19
18	Ionosphere	2	351	0	34	34
19	Iris Plant	3	150	0	4	4
20	Labor Relations	2	57	8	8	16
21	Lymphography	4	148	15	3	18
22	Mushrooms	2	8,124	22	0	22
23	Primary Tumor	22	339	17	0	17
24	Sonar	2	208	0	60	60
25	Soybean	19	683	35	0	35
26	Statlog Project: Heart Disease	2	270	7	6	13
27	Statlog Project: Vehicle Silhouettes	4	946	0	18	18
28	Thyroid Disease: Hypothyroid	4	3,772	23	7	30
29	Thyroid Disease: Sick	2	3,772	23	7	30
30	Vowel	11	990	3	10	13
31	Waveform: 5000	3	5,000	0	40	40
32	Zoo	7	101	15	2	17

k is proportional to $\log n$ on the average and approaches n in the worst case, when the generalized trees are extremely unbalanced. If the time complexity of Γ is on a higher degree than that of the base decision tree inducer (i.e., $f_\Gamma(n,m) >> f_\Psi(n,m)$), the overall training time is bounded by $\frac{(k-1)k}{2}$ times of the training time of Γ.

4 Empirical Evaluation

We have implemented the proposed algorithm for constrained cascade generalization using logistic regression and C4.5 as base inducers (i.e., $\Gamma = \{\text{logistic regression}\}$, $\Psi = \text{C4.5}$) and evaluated the implementation using 32 data sets for classification problems collected in the UCI machine learning repository [2]. These data sets have been frequently used as benchmarks to compare the performance of different classification methods in the literature. Table 1 summarizes the characteristics of these data sets. Our implementation was carried out by extending the Weka machine learning toolkit [27] in Java. The empirical evaluation was performed on a Dell Optiplex/GX260 workstation with a Pentium 4 CPU running at 2.27GHz and 512 MB RAM.

We have conducted three experiments, identifying the best cascading depth, evaluating the effects of two heuristics used in the original cascade generalization paper [12], and comparing the proposed method for constrained cascade generalization with ensemble methods, including bagging and boosting. In all the experiments, we used *stratified 10-fold cross validation*, a recommended performance estimation method [17], to estimate the accuracy of each learned classifier, starting from the same seed for the random number generator for every method in the same run. A parameter used by C4.5, the minimum number of training examples covered by a node, was uniformly set to 10. We will report on some empirical results in the following sections.

4.1 What's the Best Cascading Depth?

In the first experiment, we ran our algorithm under different $d_{cascade}$ values for each of the 32 data sets and identified the best cascading depth. Table 2 and Table 3 summarize the apparent accuracy and cross-validated accuracy of the various base or composite classifiers. Note that we use cross-validated accuracy as the performance measure in all the experiments presented in this paper and use apparent accuracy only for the purpose of explaining the internal effects of cascade generalization in this experiment. We will not present apparent accuracy results any more for the two subsequent experiments.

Apparent accuracy increases almost monotonically as $d_{cascade}$ increases, except that there are occasional fluctuations, due to the heuristic nature of C4.5. Apparent accuracy decreases only 15.9 percent (22 in 138) of the times as $d_{cascade}$ increases. Tight coupling provides the highest apparent accuracy for 87.5 percent (28 in 32) of the data sets. This confirms that the learned models tend to fit the training data better as $d_{cascade}$ increases. However, this is not true for cross-validated accuracy, which fluctuates severely without regard to apparent accuracy. Cross-validated accuracy decreases 45.7 percent (63 in 138) of the times as $d_{cascade}$

TABLE 2
Apparent Accuracy (%) for the 32 UCI Data Sets

No.	Cascading Depth											
	0	1	2	3	4	5	6	7	8	9	10	11
1	97.77	98.66	**99.78**	**99.78**	**99.78**	-	-	-	-	-	-	-
2	73.45	74.78	81.42	81.86	81.86	85.40	**85.84**	-	-	-	-	-
3	80.98	77.07	87.32	95.12	**98.54**	-	-	-	-	-	-	-
4	84.32	89.92	**91.68**	-	-	-	-	-	-	-	-	-
5	96.42	**97.57**	-	-	-	-	-	-	-	-	-	-
6	75.87	77.62	81.82	**84.62**	-	-	-	-	-	-	-	-
7	98.50	98.40	98.31	99.00	**99.59**	-	-	-	-	-	-	-
8	95.63	97.47	**97.93**	-	-	-	-	-	-	-	-	-
9	88.84	88.26	88.26	89.42	91.74	**92.46**	-	-	-	-	-	-
10	80.50	81.70	81.00	85.60	87.60	**90.90**		-	-	-	-	-
11	**82.55**	82.03	77.34	-	-	-	-	-	-	-	-	-
12	78.97	78.50	78.50	80.84	79.44	**81.31**	-	-	-	-	-	-
13	83.50	88.12	88.12	**93.73**	-	-	-	-	-	-	-	-
14	81.29	85.71	86.39	88.10	**88.78**	-	-	-	-	-	-	-
15	84.52	90.97	**98.06**	-	-	-	-	-	-	-	-	-
16	85.87	88.86	93.21	95.92	**98.37**	-	-	-	-	-	-	-
17	96.97	97.01	97.10	97.10	97.58	98.14	97.66	97.84	98.23	98.79	**99.05**	-
18	92.59	95.73	**99.15**	-	-	-	-	-	-	-	-	-
19	96.00	96.00	**98.67**	-	-	-	-	-	-	-	-	-
20	87.72	**100.00**	-	-	-	-	-	-	-	-	-	-
21	81.08	**97.97**	95.95	-	-	-	-	-	-	-	-	-
22	**100.00**	**100.00**	-	-	-	-	-	-	-	-	-	-
23	46.02	46.90	46.90	52.80	53.69	54.28	54.57	55.16	**55.46**	-	-	-
24	90.87	**100.00**	-	-	-	-	-	-	-	-	-	-
25	87.55	85.65	88.14	90.04	92.24	94.88	92.83	92.83	93.85	94.29	93.56	**95.61**
26	85.56	86.30	88.15	88.15	**90.00**	-	-	-	-	-	-	-
27	83.57	81.44	85.46	85.93	85.22	85.46	**87.47**	84.75	86.64	-	-	-
28	**99.36**	99.18	99.18	99.13	99.28	-	-	-	-	-	-	-
29	99.05	98.33	98.70	98.83	99.05	99.18	**99.23**	-	-	-	-	-
30	82.93	81.72	81.82	86.46	89.09	95.15	93.23	95.66	96.36	96.77	**97.47**	-
31	89.38	90.64	91.00	90.08	90.42	92.80	95.16	95.78	96.74	**97.00**	-	-
32	**83.17**	**83.17**	**83.17**	**83.17**	-	-	-	-	-	-	-	-

The highest apparent accuracy for each data set is bolded. A drop in apparent accuracy as $d_{cascade}$ increases is underlined.

TABLE 3
Cross-Validated Accuracy (%) for the 32 UCI Data Sets

No.	Cascading Depth											
	0	1	2	3	4	5	6	7	8	9	10	11
1	96.88	98.44	**99.22**	98.89	98.89	-	-	-	-	-	-	-
2	**69.03**	65.49	66.37	65.93	66.81	65.93	65.93	-	-	-	-	-
3	67.32	63.41	69.27	70.73	**71.71**	-	-	-	-	-	-	-
4	76.80	86.56	**90.08**	-	-	-	-	-	-	-	-	-
5	94.85	**97.00**	-	-	-	-	-	-	-	-	-	-
6	**74.83**	69.93	69.58	69.93	-	-	-	-	-	-	-	-
7	**97.97**	97.68	97.59	97.47	97.87	-	-	-	-	-	-	-
8	**95.63**	**95.63**	95.17	-	-	-	-	-	-	-	-	-
9	**86.38**	85.22	85.51	84.06	83.77	83.04	-	-	-	-	-	-
10	70.50	72.00	**72.20**	71.60	68.30	66.60	-	-	-	-	-	-
11	**75.52**	75.00	**75.52**	-	-	-	-	-	-	-	-	-
12	64.95	65.89	**67.76**	67.29	66.82	64.95	-	-	-	-	-	-
13	75.91	**83.83**	81.52	82.84	-	-	-	-	-	-	-	-
14	79.93	79.93	80.95	**81.29**	**81.29**	-	-	-	-	-	-	-
15	**84.52**	81.94	81.29	-	-	-	-	-	-	-	-	-
16	**85.87**	80.16	77.17	73.91	73.91	-	-	-	-	-	-	-
17	95.06	95.06	95.58	95.58	95.76	**95.89**	95.58	95.71	95.63	95.50	95.58	-
18	**90.60**	89.17	85.19	-	-	-	-	-	-	-	-	-
19	94.67	94.67	**96.67**	-	-	-	-	-	-	-	-	-
20	78.95	**80.70**	-	-	-	-	-	-	-	-	-	-
21	72.97	77.70	77.70	-	-			-	-	-	-	-
22	**100.00**	**100.00**	-	-	-	-	-	-	-	-	-	-
23	40.12	41.59	41.59	**45.72**	42.48	40.71	38.94	37.46	37.17	-	-	-
24	**75.48**	71.15	-	-	-	-	-	-	-	-	-	-
25	84.48	81.55	83.75	83.60	84.63	84.48	84.48	84.48	84.77	**85.65**	84.92	**85.65**
26	**83.70**	81.48	81.48	80.74	78.89	-	-	-	-	-	-	-
27	73.88	71.87	71.39	72.70	70.09	74.70	76.12	74.70	**76.24**	-	-	-
28	**99.31**	98.65	98.33	98.38	98.06	-	-	-	-	-	-	-
29	**98.59**	97.91	97.93	97.72	97.61	97.67	97.64	-	-	-	-	-
30	68.99	66.46	68.38	72.63	75.56	81.52	82.73	82.22	82.93	83.43	**83.64**	-
31	76.18	78.92	81.88	83.80	**84.08**	83.72	82.00	81.46	81.16	81.32	-	-
32	**83.17**	**83.17**	**83.17**	**83.17**	-	-	-	-	-	-	-	-

The highest cross-validated accuracy for each data set is bolded. A drop in cross-validated accuracy as $d_{cascade}$ increases is underlined.

TABLE 4
Paired *t*-tests Comparing the Proposed Method with C4.5, Loose Coupling, and Tight Coupling

Method	Sample Size	Average	StdDev	Compared to $d_{cascade}$ Tuning	
				t	Sig.
C4.5	32	81.66	13.08	3.60	0.001
Loose Coupling	32	81.51	13.32	4.23	0.000
Tight Coupling	32	82.03	13.43	3.97	0.000
$d_{cascade}$ Tuning	32	84.06	11.94		

TABLE 5
Training Time for the 32 UCI Data Sets

No.	#Trees k	Training Time (Seconds)				Inequality (8) holds	Error (%)
		Logistic Regression f_Γ	C4.5 f_Ψ	Cascade f_{trees}	$\frac{(k-1)k}{2} f_\Gamma + k f_\Psi$		
1	5	5.78	0.36	42.13	59.60	Yes	
2	7	4.92	0.27	109.87	105.21	No	4.24
3	5	3.03	0.14	22.39	31.00	Yes	
4	3	0.64	0.09	2.05	2.19	Yes	
5	2	0.58	0.09	0.73	0.76	Yes	
6	4	1.08	0.05	6.16	6.68	Yes	
7	5	47.14	0.38	235.27	473.30	Yes	
8	3	0.73	0.06	2.22	2.37	Yes	
9	6	2.53	0.13	42.02	38.73	No	7.83
10	6	3.02	0.17	65.84	46.32	No	29.65
11	3	0.47	0.13	1.68	1.80	Yes	
12	6	2.25	0.14	30.20	34.59	Yes	
13	4	0.88	0.09	6.71	5.64	No	15.95
14	5	0.86	0.08	8.94	9.00	Yes	
15	3	0.45	0.08	1.35	1.59	Yes	
16	5	2.13	0.09	17.80	21.75	Yes	
17	11	50.45	0.89	1548.50	2784.54	Yes	
18	3	1.38	0.23	3.75	4.83	Yes	
19	3	0.63	0.05	1.74	2.04	Yes	
20	2	0.30	0.03	0.36	0.36	Yes	
21	3	0.63	0.06	1.87	2.07	Yes	
22	2	265.51	0.31	281.11	266.13	No	5.33
23	9	25.03	0.19	680.91	902.79	Yes	
24	2	0.72	0.20	1.04	1.12	Yes	
25	12	113.81	0.45	3260.47	7516.86	Yes	
26	5	0.36	0.08	3.77	4.00	Yes	
27	9	13.44	0.28	243.12	486.36	Yes	
28	5	348.55	0.50	2127.00	3488.00	Yes	
29	7	31.60	0.69	563.28	668.43	Yes	
30	11	66.05	0.59	1852.57	3639.24	Yes	
31	10	88.42	4.50	2611.11	4023.90	Yes	
32	4	1.19	0.06	6.30	7.38	Yes	

increases. Tight coupling provides the highest cross-validated accuracy for only 37.5 percent (12 in 32) of the data sets. This is consistent with our proposition that cascade generalization is subject to the complexity-generalizability trade off, just like any other classification scheme. Neither loose coupling nor tight coupling, but some moderate coupling tuned to each particular application, is the best cascading strategy.

The best $d_{cascade}$ in terms of cross-validated accuracy can be found for each data set. Paired *t*-tests (summarized in Table 4) show that the proposed method based on $d_{cascade}$ tuning significantly improves accuracy of C4.5 ($t(31) = 3.60$, $p = 0.001$) and is also significantly superior to both loose coupling ($t(31) = 4.23$, $p = 0.000$) and tight coupling ($t(31) = 3.97$, $p = 0.000$) on the average.

Table 5 summarizes the training time of the various base or composite classification methods. Inequality (8) holds for most of the data sets, except for five data sets, where f_{trees} is over the predicted bound slightly (within 30 percent). Testing time is negligible, compared to training time, for all the methods and is not evaluated in this paper.

4.2 Visualizing Why Cascade Generalization Works

The central theme of our proposed algorithm for constrained cascade generalization is to search for an appropriate level of complexity fit so that the learned model fits the training data well to an extent where it also generalizes well to unseen data. The idea can be better understood by visualizing the models learned by different methods in a two-dimensional feature space. Therefore, we ran different methods for the "Credit Card Approval" data set using two continuous features, A8 and A11.

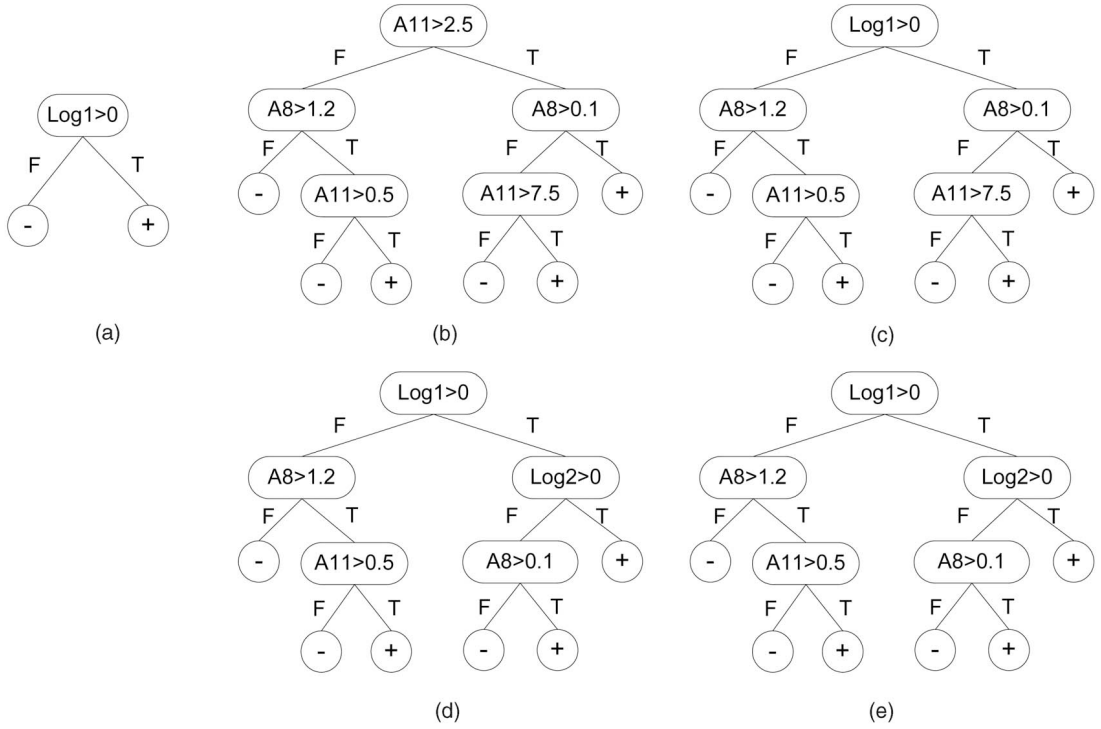


Fig. 1. Classification models learned by different classification methods for the "credit card approval" data set using two features. $Log1 = 0.21 * A8 + 0.37 * A11 - 1.30$, $Log2 = 0.37 * A8 + 0.09 * A11 - 0.40$. (a) Logistic regression. (b) C4.5. (c)-(e) Cascading logistic regression and C4.5, $d_{cascade} = 1 - 3$.

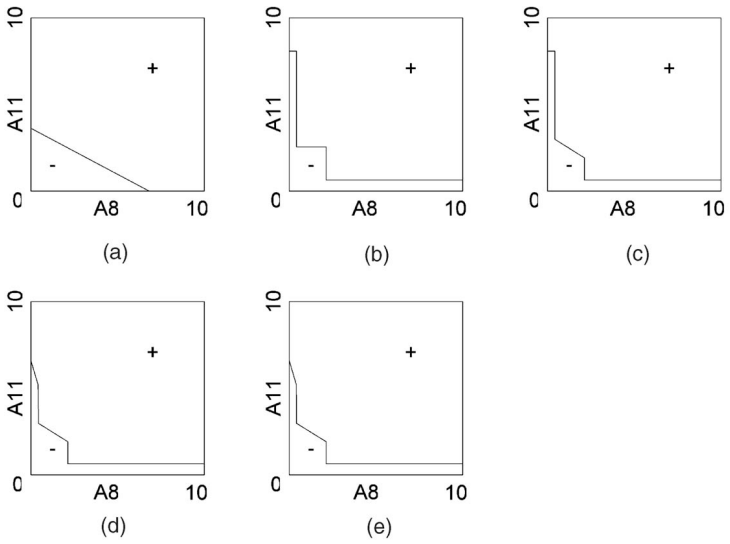


Fig. 2. Decision boundaries generated by different classification methods for the "credit card approval" data set using two features. (a) Logistic regression. (b) C4.5. (c)-(e) Cascading logistic regression and C4.5, $d_{cascade} = 1 - 3$.

The learned models are presented in Fig. 1. The decision boundaries of the models are illustrated in Fig. 2. Logistic regression (Fig. 2a) finds a single line to separate the two classes. Decision tree (Fig. 2b) uses a sequence of line segments to approximate the boundary between the two classes; all of the line segments are restricted to be parallel to one of the two axes. Cascading logistic regression with decision tree (Fig. 2c, Fig. 2d, and Fig. 2e) allows some of the line segments to be oblique; the directions of these oblique lines are determined by the cascaded method, logistic regression. The number of such oblique lines is controlled by the parameter $d_{cascade}$. The larger $d_{cascade}$ is, the more oblique lines are allowed, the more flexible the model is. However, when there are too many such oblique lines, those chosen later in the process are based on only a few unbalanced examples and become unreliable, i.e., they describe the characteristics of the few training examples, but may not represent the entire *population* well. In this example, $d_{cascade} = 1$ corresponds to loose coupling and $d_{cascade} = 3$ corresponds to tight coupling.

There are only four trees in this simple example. The cascaded logistic regression models were not actually selected at some internal tree nodes by the heuristic

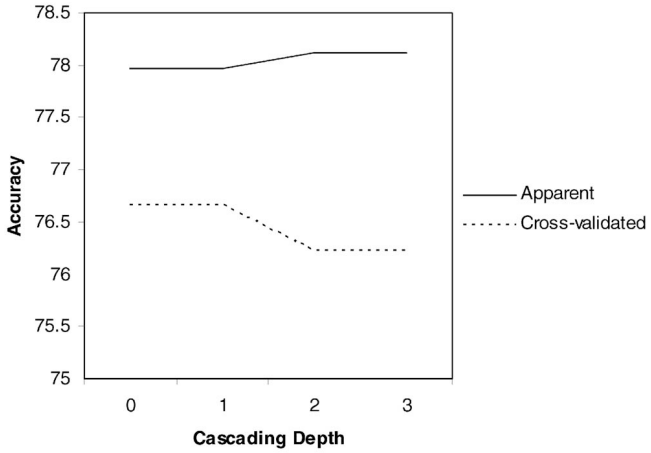


Fig. 3. Accuracy of generalized decision trees with different $d_{cascade}$'s.

splitting mechanism of C4.5. The tree with $d_{cascade} = 2$ and the tree with $d_{cascade} = 3$ are identical. There will be more trees and more cascaded models in a more complex situation. The best $d_{cascade}$ needs to be determined empirically for a particular application.

Fig. 3 shows the trends of apparent accuracy and cross-validated accuracy as $d_{cascade}$ increases. Apparent accuracy monotonically increases as $d_{cascade}$ increases, indicating that the models fit the training data better and better. However, cross-validated accuracy goes down, indicating that overfitting occurred. In this application, the best $d_{cascade}$ is 0, i.e., no cascading at all, if only the two features A8 and A11 are used for classification.

4.3 Effects of Additional Heuristics

The original paper on cascade generalization [12] adopted two heuristics to restrict cascading: 1) A constructed feature corresponding to class i is used only if the number of examples, at the current node, belonging to class i is greater than Nm, where m is the number of features and N is 3 by default. 2) The error rate of the cascaded classifier should be less than 0.5 in the training data for the classifier to be used. The results presented in Section 4.1 were generated without using these heuristics. We have also evaluated the effects of these heuristics in conjunction with our proposed method based on $d_{cascade}$ tuning using a factorial experiment design shown in Table 6. The cross-validated accuracy of the various learned classifiers is summarized in Table 7. We use the GLM Repeated Measures procedure for analysis of variance (ANOVA) in SPSS to test the effects of the factors. This procedure provides ANOVA when the same measurement is made several times on each case. The test shows the following results (summarized in Table 8):

1. $d_{cascade}$ tuning significantly improves accuracy ($F(1, 31) = 22.11$, $p = 0.000$).
2. Heuristic 1 has a negative effect on accuracy ($F(1, 31) = 7.32$, $p = 0.011$).
3. The effect of heuristic 2 is not significant ($F(1, 31) = 1.17$, $p = 0.288$).
4. The interaction between $d_{cascade}$ tuning and heuristic 1 is significant ($F(1, 31) = 5.98$, $p = 0.020$). Applying heuristic 1 reduces the effect of $d_{cascade}$ tuning.

TABLE 6
Experiment Design: Effects of $d_{cascade}$ Tuning and Two Heuristics

Setting	$d_{cascade}$ Tuning	Heuristic 1	Heuristic 2
1	No	No	No
2	No	No	Yes
3	No	Yes	No
4	No	Yes	Yes
5	Yes	No	No
6	Yes	No	Yes
7	Yes	Yes	No
8	Yes	Yes	Yes

5. There is no significant interaction between $d_{cascade}$ tuning and heuristic 2 ($F(1, 31) = 1.42$, $p = 0.243$).
6. The two heuristics significantly interfere with each other ($F(1, 31) = 5.24$, $p = 0.029$).
7. There is no significant interaction among the three factors simultaneously ($F(1, 31) = 1.85$, $p = 0.184$).

4.4 Comparison with Bagging and Boosting

We have also compared the proposed algorithm for constrained cascade generalization with bagging [4] and AdaBoost [10]. The two heuristics evaluated in the previous section were not used in cascade generalization. For bagging, 10 classifiers generated by C4.5 were bagged. For AdaBoost, up to 10 classifiers generated by C4.5 were boosted. Table 9 summarizes the cross-validated accuracy generated by the various methods. While the proposed algorithm is marginally superior to both bagging and AdaBoost, paired *t*-tests do not show significant differences between the methods, due to the small sample size (32); comparing cascading and bagging: $t(31) = 1.24$, $p = 0.224$; comparing cascading and boosting: $t(31) = 0.131$, $p = 0.897$.

4.5 Discussion

While we have obtained some statistically significant results in our empirical evaluation, these results should be interpreted with caution, taking the following issues in consideration:

1. Statistical significance should not be confused with practical significance. The fact that the effect of some technique is statistically significant shows that the effect is very likely to be genuine rather than by chance, but does not tell whether the effect is practically important, large, or even useful in the context. Interpretation of practical significance is necessarily domain dependent. For example, in our first experiment, cascading logistic regression reduced the error rate of C4.5 by 2.4 percent on the average. An error reduction of this size can be conceived as practically significant for some, but not all, classification problems.
2. Since most of the empirical studies on supervised learning in the literature, including this one, use conveniently available data sets, such as those in the UCI repository, rather than truly random samples, the generalizability of the evaluation results is dependent on the representativeness of these data sets. In addition, our evaluation compares the average performance of different methods across

TABLE 7
Cross-Validated Accuracy under $d_{cascade}$ Tuning and Two Heuristics

No	Experiment Setting							
	1	2	3	4	5	6	7	8
1	98.89	98.89	97.77	97.77	99.22	99.22	97.77	97.77
2	65.93	65.93	65.93	65.93	69.03	66.81	69.03	69.03
3	71.71	71.71	63.41	63.41	71.71	71.71	69.27	69.27
4	90.08	90.08	90.08	90.08	90.08	90.08	90.08	90.08
5	97.00	97.00	97.00	97.00	97.00	97.00	97.00	97.00
6	69.93	68.88	68.88	68.88	74.83	69.93	74.83	69.93
7	97.87	97.81	97.81	97.81	97.97	97.87	97.97	97.87
8	95.17	95.17	95.17	95.17	95.63	95.63	95.63	95.63
9	83.04	83.04	83.04	83.04	86.38	85.51	86.38	85.51
10	66.60	66.60	66.60	66.60	72.20	72.20	72.20	72.20
11	75.52	75.00	75.00	75.00	75.52	75.52	75.52	75.52
12	64.95	64.95	64.49	64.49	67.76	67.76	66.82	66.82
13	82.84	81.52	81.52	81.52	83.83	83.83	83.83	83.83
14	81.29	80.95	80.95	80.95	81.29	81.29	81.29	81.29
15	81.29	80.65	77.42	77.42	84.52	81.94	84.52	84.52
16	73.91	73.91	73.91	73.91	85.87	80.16	85.87	80.16
17	95.58	95.54	95.54	95.54	95.89	95.89	95.89	95.89
18	85.19	85.19	85.19	85.19	90.60	89.17	90.60	89.17
19	96.67	96.67	96.67	96.67	96.67	96.67	96.67	96.67
20	80.70	80.70	66.67	66.67	80.70	80.70	78.95	78.95
21	77.70	77.70	77.70	77.70	77.70	77.70	77.70	77.70
22	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
23	37.17	37.17	39.23	39.23	45.72	45.72	42.18	42.18
24	71.15	71.15	75.00	75.00	75.48	71.15	75.48	75.48
25	85.65	85.80	87.99	87.99	85.65	85.80	88.14	88.14
26	78.89	78.52	78.52	78.52	83.70	81.48	83.70	83.70
27	76.24	76.12	76.24	76.12	76.24	76.24	76.24	76.24
28	98.06	97.96	97.99	97.99	99.31	98.65	99.31	98.65
29	97.67	97.64	97.64	97.64	98.59	97.93	98.59	97.93
30	83.64	83.64	83.64	83.64	83.64	83.64	83.64	83.64
31	81.32	82.40	82.40	82.40	84.08	84.08	84.08	84.08
32	83.17	83.17	83.17	83.17	83.17	83.17	83.17	83.17
Average	82.03	81.92	81.33	81.33	84.06	83.26	83.82	83.38
StdDev	13.43	13.47	13.74	13.74	11.94	12.23	12.43	12.49

TABLE 8
Results of the GLM Repeated Measures Procedure for ANOVA in SPSS

Factor(s)	F	Hypothesis df	Error df	Sig.
$d_{cascade}$ Tuning	22.11	1	31	0.000
Heuristic1	7.32	1	31	0.011
Heuristic2	1.17	1	31	0.288
$d_{cascade}$ Tuning * Heuristic1	5.98	1	31	0.020
$d_{cascade}$ Tuning * Heuristic2	1.42	1	31	0.243
Heuristic1 * Heuristic2	5.24	1	31	0.029
$d_{cascade}$ Tuning * Heuristic1 * Heuristic2	1.85	1	31	0.184

classification problems, but cannot be used to predict which method will be superior to others for a particular problem. No method has been found so far that is universally superior to others in all problems; indeed, we believe, based on the "*no-free-lunch theorems*" [28], that it's likely such a method does not exist. In practice, for a given new classification problem, various methods need to be empirically evaluated to find the best ones, with previous empirical results as guidelines.

3. Note that considering the relatively small sizes of the data sets, we have selected a relatively small value (10) for the number of iterations in bagging and boosting. The accuracy and training times of bagging and boosting are dependent on this parameter. A more elaborate comparison of accuracy and training times of the methods should take the effect of this parameter into account.
4. Note that the potential gain in classification accuracy provided by the proposed algorithm does not come for free. As constructive features (often linear combinations of original features) are incorporated into decision trees, the comprehensibility of decision trees degrades. The linear combinations of multiple features in a multivariate tree are harder to interpret than the individual features in a univariate tree. Indeed, any method that attempts to combine multiple classifiers for potential performance improvement, including bagging and boosting, will inevitably reduce the comprehensibility of learned models. Accuracy and comprehensibility are unavoidable trade-offs; different methods need to be evaluated for a particular classification problem to find appropriate ones. A large reason that decision tree techniques have been popular in practice is that they generate simple and easily interpretable models. In situations where comprehensibility is

TABLE 9
Cross-Validated Accuracy of Cascading, Bagging, and Boosting

No	Cascading	Bagging	Boosting
1	99.22	96.77	99.55
2	69.03	68.14	70.80
3	71.71	72.20	74.15
4	90.08	82.72	83.36
5	97.00	94.99	95.71
6	74.83	73.78	69.93
7	97.97	98.31	99.56
8	95.63	95.86	96.09
9	86.38	85.94	83.77
10	72.20	75.10	70.00
11	75.52	75.78	73.96
12	67.76	72.43	73.83
13	83.83	78.88	78.55
14	81.29	80.61	79.93
15	84.52	79.35	83.23
16	85.87	85.60	83.70
17	95.89	95.84	98.01
18	90.60	90.88	91.74
19	96.67	94.67	92.67
20	80.70	84.21	78.95
21	77.70	80.41	81.08
22	100.00	100.00	100.00
23	45.72	42.18	40.12
24	75.48	77.88	79.81
25	85.65	89.17	93.12
26	83.70	81.85	78.52
27	76.24	75.30	76.48
28	99.31	99.31	99.63
29	98.59	98.44	98.78
30	83.64	80.00	89.49
31	84.08	81.68	80.54
32	83.17	83.17	92.08
Average	84.06	83.48	83.97
StdDev	11.94	11.90	12.69

weighted higher than accuracy, simple univariate trees may well be preferred to more complex models, even if the complex models are more accurate.

5 Conclusions and Future Research

In this paper, we have proposed a generic algorithm of constrained cascade generalization, of which cascading methods proposed in the past, including loose coupling and tight coupling, are strictly special cases. We have empirically evaluated the algorithm in a number of classification problems. Empirical results show that loose coupling and tight coupling are not always the best cascading strategies and that the maximum cascading depth parameter in the proposed algorithm can be tuned for better classification accuracy. We have also found that the proposed algorithm is marginally better than bagging and boosting on the average.

While we have proposed one approach to constraining the degree of cascade generalization and empirically evaluated a particular cascade generalization using C4.5 and logistic regression as base inducers, there are still many issues related to cascade generalization that need to be studied in future research. Some examples are:

1. All data sets used in our current empirical evaluation are relatively small. A study on the scalability of the proposed algorithm in large data sets is needed.
2. Some advantages of univariate decision trees, besides their relatively high comprehensibility, include that they are invariant to monotone transformations of the input variables and can deal with missing values and outliers. These aspects need to be analyzed in the context of multivariate decision trees.
3. In our current algorithm, we use the maximum cascading depth as a parameter to constrain cascading during tree induction. There are other potential parameters, such as statistical significance tests of learned model, to constrain cascading. In addition, these parameters are related to tree pruning parameters. The effects of different parameters and the interactions among them need to be further analyzed and evaluated.
4. Currently, we select the best tree (i.e., an *elitist* strategy) from a set of trees with different $d_{cascade}$'s. An alternative strategy is to combine the multiple trees via a "*voting*" mechanism (i.e., an *ensemble* strategy). Intuitively, this strategy can potentially reduce both bias (due to cascade generalization) and variance (due to voting) and provide more performance gain in particular situations. The two strategies need to be analytically and empirically compared.
5. Cascading decision tree inducers with nonlinear model inducers, such as Naive Bayes and artificial neural networks, needs to be investigated.
6. We developed the proposed algorithm in our research in *entity identification* [24], [30] (also called *record linkage* [9]), i.e., identifying records that semantically correspond to the same entity in the real world from different data sources. We are evaluating the algorithm in this problem domain.

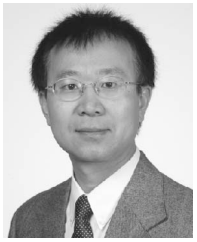
Acknowledgments

The authors are grateful to Associate Editor, Dr. Paolo Frasconi, and the three anonymous reviewers for their many valuable suggestions.

References

[1] E. Bauer and R. Kohavi, "An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants," *Machine Learning*, vol. 36, nos. 1-2, pp. 105-139, 1999.
[2] C.L. Blake and C.J. Merz, UCI Repository of Machine Learning Databases, http://www.ics.uci.edu/~mlearn/MLRepository.html, 1998.
[3] J.C. Bioch, O. van der Meer, and R. Potharst, "Bivariate Decision Trees," *Principles of Data Mining and Knowledge Discovery, Lecture Notes in Artificial Intelligence 1263*, J. Komorowski and J. Zytkow, eds., Springer Verlag, pp. 232-243, 1997.
[4] L. Breiman, "Bagging Predictors," *Machine Learning*, vol. 24, no. 2, pp.123-140, 1996.
[5] C.E. Brodley and P.E. Utgoff, "Multivariate Decision Trees," *Machine Learning*, vol. 19, no. 1, pp. 45-77, 1995.
[6] T.G. Dietterich, "Ensemble Methods in Machine Learning," *Proc. First Int'l Workshop Multiple Classifier Systems*, pp. 1-15, 2000.
[7] T.G. Dietterich, "An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization," *Machine Learning*, vol. 40, no. 2, pp. 139-157, 2000.
[8] P. Domingos, "A Unified Bias-Variance Decomposition and its Applications," *Proc. 17th Int'l Conf. Machine Learning*, pp. 231-238, 2000.
[9] I.P. Fellegi and A.B. Sunter, "A Theory of Record Linkage," *J. American Statistical Assoc.*, vol. 64, pp. 1183-1210, 1969.
[10] Y. Freund and R.E. Schapire, "Experiments with a New Boosting Algorithm," *Proc. 13th Int'l Conf. Machine Learning*, pp. 148-156, 1996.

[11] J. Gama, "Discriminant Trees," *Proc. 16th Int'l Conf. Machine Learning,* pp. 134-142, 1999.
[12] J. Gama and P. Brazdil, "Cascade Generalization," *Machine Learning,* vol. 41, no. 3, pp. 315-343, 2000.
[13] S. Geman, E. Bienenstock, and R. Doursat, "Neural Networks and the Bias/Variance Dilemma," *Neural Computation,* vol. 4, pp. 1-48, 1992.
[14] D. Heath, S. Kasif, and S. Salzberg, "Induction of Oblique Decision Trees," *Proc. 13th Int'l Joint Conf. Artificial Intelligence,* pp. 1002-1007, 1993.
[15] D.W. Hosmer and S. Lemeshow, *Applied Logistic Regression,* second ed. John Wiley & Sons, Inc., 2000.
[16] G.H. John, "Robust Linear Discriminant Trees," *Learning From Data: Artificial Intelligence and Statistics V, Lecture Notes in Statistics,* D. Fisher and H. Lenz, eds., Springer-Verlag, pp. 375-385, 1996.
[17] R. Kohavi, "A Study of Cross-validation and Bootstrap for Accuracy Estimation and Model Selection," *Proc. 14th Int'l Joint Conf. Artificial Intelligence,* pp. 1137-1143, 1995.
[18] R. Kohavi and D.H. Wolpert, "Bias Plus Variance Decomposition for Zero-One Loss Functions," *Proc. 13th Int'l Conf. Machine Learning,* pp. 275-283, 1996.
[19] S.K. Murthy, "Automatic Construction of Decision Trees from Data: A Multi-Disciplinary Survey," *Data Mining and Knowledge Discovery,* vol. 2, no. 4, pp. 345-389, 1998.
[20] S.K. Murthy, S. Kasif, and S. Salzberg, "A System for Induction of Oblique Decision Trees," *J. Artificial Intelligence Research,* vol. 2, pp. 1-32, 1994.
[21] J.R. Quinlan, "Discovering Rules by Induction from Large Collections of Examples," *Expert Systems in the Micro Electronic Age,* D. Michie, ed., Edinburgh Univ. Press, 1979.
[22] J.R. Quinlan, "Induction of Decision Trees," *Machine Learning,* vol. 1, no. 1, pp. 81-106, 1986.
[23] J.R. Quinlan, *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1993.
[24] S. Ram and H. Zhao, "Detecting Both Schema-Level and Instance-Level Correspondences for the Integration of E-Catalogs," *Proc. 11th Ann. Workshop Information Technology and Systems (WITS '01),* pp. 193-198, 2001.
[25] S.M. Weiss and C.A. Kulikowski, *Computer Systems That Learn—Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert System.* Morgan Kaufmann, 1991.
[26] J. Wickramaratna, S. Holden, and B. Buxton, "Performance Degradation in Boosting," *Proc. Second Int'l Workshop Multiple Classifier Systems,* pp. 11-21, 2001.
[27] I.H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementation.* Morgan Kaufmann, 2000.
[28] D.H. Wolpert, "The Relationship Between PAC, the Statistical Physics Framework, the Bayesian Framework, and the VC Framework," *The Mathematics of Generalization—Proc. SFI/CNLS Workshop Formal Approaches to Supervised Learning,* pp. 117-214, D.H. Wolpert, ed., Addison-Wesley, 1994.
[29] O.T. Yildiz and E. Alpaydin, "Linear Discriminant Trees," *Proc. 17th Int'l Conf. Machine Learning,* pp. 1175-1182, 2000.
[30] H. Zhao and S. Ram, "Entity Identification for Heterogeneous Database Integration—A Multiple Classifier System Approach and Empirical Evaluation," *Information Systems,* 2004.



Huimin Zhao received the BE and ME degrees in automation from Tsinghua University, China, in 1990 and 1993, respectively, and the PhD degree in management information systems from the University of Arizona in 2002. He is assistant professor of management information systems in the School of Business Administration at the University of Wisconsin–Milwaukee. His current research interests include data mining, data integration, and Web services. He has published in such journals as *Information Systems* and the *International Journal of Web Services Research*. He serves on the editorial board of the *International Journal for Infonomics* and the editorial review board of the *Journal of Database Management*. He is a member of the IEEE, the Association for Information Systems (AIS), and the Information Resources Management Association (IRMA).

Sudha Ram received the BS degree in mathematics, physics, and chemistry from the University of Madras in 1979, the PGDM from the Indian Institute of Management, Calcutta, in 1981, and the PhD degree from the University of Illinois at Urbana-Champaign, in 1985. She is an Eller Professor of management information systems in the College of Business and Public Administration at the University of Arizona. Dr. Ram has published articles in such journals as *Communications of the ACM*, *IEEE Expert*, the *IEEE Transactions on Knowledge and Data Engineering*, *Information Systems*, *Information Systems Research*, *Management Science*, and *MIS Quarterly*. Dr. Ram's research deals with issues related to enterprise data management. Her research has been funded by organizations such as IBM, Intel Corporation, Raytheon, US ARMY, National Institute of Science and Technology, US National Science Foundation, NASA, and the Office of Research and Development at the CIA. Specifically, her research deals with interoperability among heterogeneous database systems, semantic modeling, bioinformatics and spatio-temporal semantics, business rules modeling, Web services discovery and selection, and automated software tools for database design. Dr. Ram serves on editorial board of such journals as *Decision Support Systems*, *Information Systems Frontiers*, *Journal of Information Technology and Management*, and as associate editor for *Information Systems Research*, the *Journal of Database Management*, and the *Journal of Systems and Software*. She has chaired several workshops and conferences supported by the ACM, the IEEE, the IEEE Computer Society, and the AIS. She is a cofounder of the Workshop on Information Technology and Systems (WITS) and serves on the steering committee of many workshops and conferences including the Entity Relationship Conference (ER). Dr. Ram is a member of the ACM, the IEEE, the IEEE Computer Society, INFORMS, and the Association for Information Systems (AIS). She is also the director of the Advanced Database Research Group based at the University of Arizona.