Data Mining and Knowledge Discovery, 6, 361–391, 2002

High-Performance Commercial Data Mining: A Multistrategy Machine Learning Application

WILLIAM H. HSU bhsu@cis.ksu.edu
Department of Computing and Information Sciences, Kansas State University, Manhattan, KS 66506;
Automated Learning Group, National Center for Supercomputing Applications (NCSA), Champaign, IL 61820

MICHAEL WELGE welge@ncsa.uiuc.edu
TOM REDMAN redman@ncsa.uiuc.edu
DAVID CLUTTER clutter@ncsa.uiuc.edu
Automated Learning Group, National Center for Supercomputing Applications (NCSA), Champaign, IL 61820

Editors: Fayyad, Mannila, Ramakrishnan

Received July 7, 2000; Revised October 10, 2001

Abstract. We present an application of inductive concept learning and interactive visualization techniques to a large-scale commercial data mining project. This paper focuses on design and configuration of high-level optimization systems (wrappers) for relevance determination and constructive induction, and on integrating these wrappers with elicited knowledge on attribute relevance and synthesis. In particular, we discuss decision support issues for the application (cost prediction for automobile insurance markets in several states) and report experiments using *D2K*, a Java-based visual programming system for data mining and information visualization, and several commercial and research tools. We describe exploratory clustering, descriptive statistics, and supervised decision tree learning in this application, focusing on a parallel genetic algorithm (GA) system, *Jenesis*, which is used to implement relevance determination (attribute subset selection). Deployed on several high-performance network-of-workstation systems (Beowulf clusters), *Jenesis* achieves a linear speedup, due to a high degree of task parallelism. Its test set accuracy is significantly higher than that of decision tree inducers alone and is comparable to that of the best extant search-space based wrappers.

Keywords: constructive induction, scalable high-performance computing, real-world decision support applications, relevance determination, genetic algorithms, software development environments for knowledge discovery in databases (KDD)

1. Introduction

This paper discusses a commercial decision support project where data mining techniques are applied to a large customer database. It presents these techniques—data cleaning, quantization, exploratory analysis (dimensionality reduction and descriptive statistics), supervised inductive learning, attribute subset selection, and interactive visualization—in a survey of the project life cycle. Concurrently, this survey presents components of a *rapid application development* environment for high-performance knowledge discovery in databases (KDD), called *D2K*. We describe a KDD process specification and its implementation using *D2K* and several commercial and experimental data mining packages. Our focus is *Jenesis*, a parallel genetic algorithm (GA) that implements and calibrates a wrapper for attribute subset selection, cf. Kohavi and John (1997). During the implementation of our experimental data mining system for the decision support application, interactive (visualization-driven) and automated methods for constructive induction were used, which drove the development of the GA, and which provided prior knowledge for the optimization subproblem (selecting and synthesizing relevant attributes). We report on the deployment of the GA in high-performance network-of-workstation environments: its performance on two Beowulf clusters is compared to that of the *MLC++* wrapper for feature subset selection (Kohavi and Sommerfield, 1996). Experimental evaluation uses the refined data set from the commercial decision support problem. The paper concludes with an account of system deployment: the methodology and process of its delivery to users, reuse issues in *D2K*, and the impact of the project results as a decision support resource.

1.1. Framework: High-performance KDD for prediction and decision support

We begin with a description of the NCSA *Data to Knowledge* (*D2K*) system and the research objectives that guide its development. *D2K* comprises a visual programming system and a scalable framework for implementing wrappers for performance enhancement in inductive learning. Written in Java, it uses *JavaDoc* for literate programming, and provides a specification mechanism for data flow in a prototypical KDD process, as depicted in figure 1.

The key novel contributions of the system are:

1. The explicit organization of learning components into *recombinable and reusable classes*
2. An *interactive approach to constructive induction* based on visualization, descriptive statistics, and preliminary data clustering
3. *Trainable hyperparameters* for bias optimization (change of representation, technique selection for overfitting prevention and avoidance)

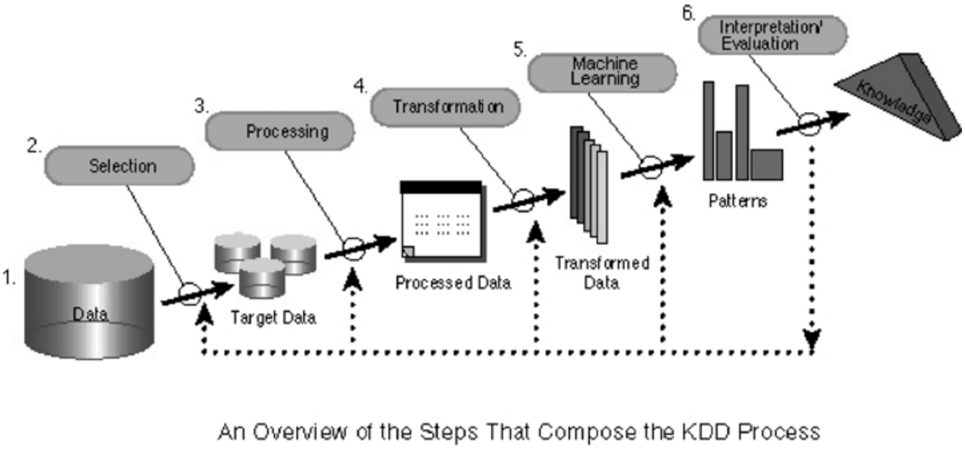

Figure 1. Data flow in a prototypical KDD application (adapted from Fayyad et al. (1996)).

4. A *hierarchical genetic algorithm* (implemented using an efficient distributed, parallel system) for attribute subset selection and reduced-error pruning that calibrates and uses these hyperparameters

Control of model complexity to achieve higher test set accuracy (through control of overfitting) is the key design goal of *D2K* that is documented in this paper. Typical applications for a system are decision support problems that are susceptible to overfitting due to many irrelevant attributes (John et al., 1994), or (usually worse) redundancy among attribute subsets (Kohavi and John, 1997).

The rest of this section outlines the abstract application (decision support in uncertain optimization domains), the isolation of a KDD problem (predictive classification) from the representative project and several similar industrial projects at NCSA, and the role of *D2K* and its high-performance computing platform.

1.1.1. KDD application: Risk prediction for decision support. The significance of data mining in many decision support problems posed to NCSA through its Industrial Partners Program is often due to the need for a predictive model for policy optimization. In some cases, this requires a performance element that solves a constraint system (e.g., adaptive dynamic programming, or ADP, methods such as value iteration and policy iteration (Russell and Norvig, 1995); in others, such as the auto insurance underwriting application described in this paper, decisions are coarser-grained or more qualitative. The decision support objective of the auto insurance underwriting application is to answer the following questions, formulated in order:

1. Do the attributes currently used to classify customers in the existing rule-based system carry any *predictive information* for the analytical objective (prediction of paid loss)?
2. Can this objective be used to define a family of *discrete supervised learning problems* (such that inductive learning can produce a model with *discriminatory power* among classes)?
3. If so, what *data integrity* issues affect this learning problem with respect to the objective?
4. What subsets of attributes (selected and synthetic) are *most relevant* to the objective?
5. What *level of accuracy* in predicting the objective yields useful decision support power?

D2K has also been applied to decision support problems in text report mining (document categorization and tracking for detection of emerging issues in quality control of truck engines) and resource allocation (prediction and monitoring of home repair demand by service category for short and long term ADP-based optimization) (Hsu and Welge, to appear).

1.1.2. Data management and exploration: Research objectives. Three issues motivate the design of the *D2K* components that are documented in this paper: control of overfitting through relevance determination, scalability, and validation methods for the performance tuning wrappers themselves. First, we address control of overfitting by applying wrappers for *moderated* attribute subset selection. The moderating criteria depend on the type of model (hypothesis representation) used and its description length—e.g., decision trees and number of decision nodes—and the importance of minimizing the subset of relevant attributes.

Second, we address scalability through our configuration of high-performance platforms for *D2K* and parallel, distributed implementations of the search criteria—namely, genetic wrappers (Cherkauer and Shavlik, 1996; Raymer et al., 1997; Hsu et al., 1999) Third, we address *meta-learning*, or *bias optimization*, through abstraction (variabilization) of the wrapper "constants" and calibration of these variables, or hyperparameters, through further statistical validation.

The role of high-performance computing is twofold: first, a distributed, shared memory (DSM) system is ideal for the application, because of the high degree of functional (task-level) parallelism in the genetic wrappers used. Since the experimental focus in *D2K* was on attribute subset selection and high-level validation of the selection criteria, we found that the symmetric multiprocessing (SMP) model also yielded performance gains and high scalability. The high performance-to-price ratio of commodity off-the-shelf (COTS) systems, such as the multiprocessor Beowulf clusters we report on in this paper, makes them an appropriate choice for implementing this DSM model.

1.2. Solution approach: D2K

Figure 2 illustrates the design of *D2K* (Auvil et al., 1999), a rapid application development system for high-performance (parallel and distributed) KDD. *D2K* is a visual programming system that manages multiple knowledge sources and provides a standardized application

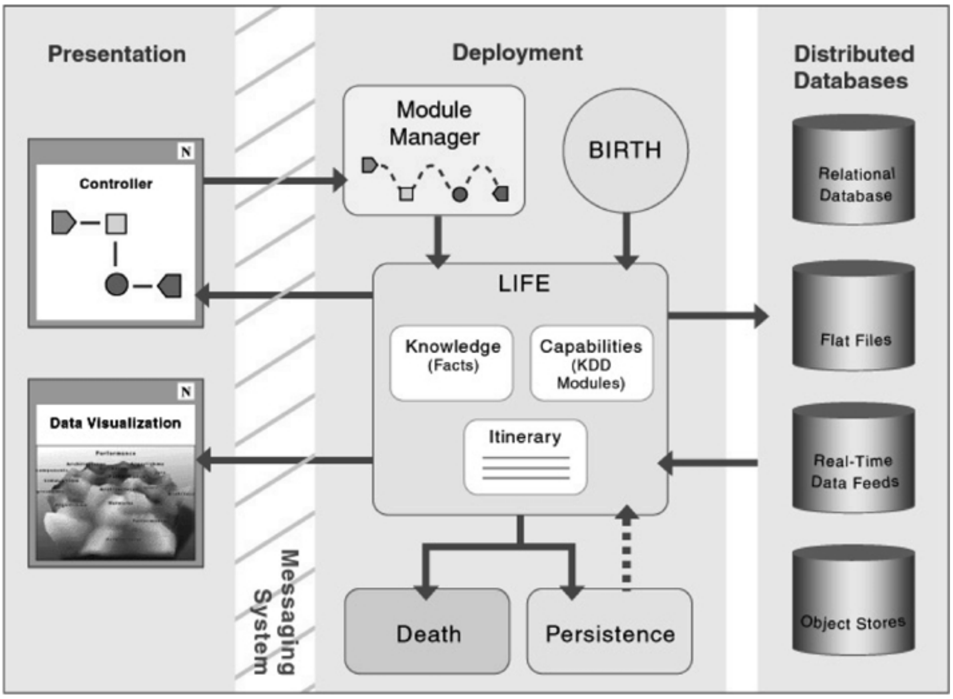

Figure 2. *D2K*: A rapid application development system for KDD.

programmer interface (API) for learning components. It implements the data mining life cycle through persistence of model representations (e.g., serialization of trained inducers for performance tuning wrappers and committee machines (Kohavi and Sommerfield, 1996; Haykin, 1999). The KDD process is observed and controlled through a presentation layer that acquires user specifications and delivers interactive visualization and output post-processing functions. Benefits of the *D2K* system include reuse, modularity, task-level parallelism, and distributivity, which are documented in Auvil et al. (1999) and surveyed for this application.

1.2.1. Role of supervised learning, data clustering, and constructive induction wrappers. Supervised learning for the auto insurance underwriting problem is formulated as a classification problem, but we also defined a data mining problem that uses inductive learning to predict migration of instances (insurance policy records) between classes. The classification model was used to audit an existing rule-based classification system over the same instance space, and to calibrate an underwriting model (to guide pricing decisions for policies) for an experimental market. The migration model was designed to track *churn* between the user's indemnity and non-indemnity divisions, and can be applied (through optimization by ADP) to generate recommendations for distributing *tiers* (classes) in the experimental market, subject to the objective of equalizing expected cost across classes. The inducer (supervised inductive learning algorithm) used for this project was *ID3* (Quinlan, 1985), but other inducers are used in this same framework: simple (*aka* naïve) Bayes for the text mining (story report) project, feedforward artificial neural networks (ANNs) and hidden Markov models (HMMs) for the demand prediction project (Hsu and Welge, to appear).

Unsupervised learning fills a twofold exploratory role in this project: first, as an technique for dimensionality reduction, or vector quantization, in order to compute descriptive statistics; second, as a mapping from raw data to synthetic classes to guide *rule refinement* in the original knowledge-based classification system. This *cluster definition* step can also be used to perform *change of representation* (Benjamin, 1990; Donoho, 1996) for the supervised classifier learning stage.

Both cluster definition and feature extraction and construction (i.e., dimensionality-reducing transforms, attribute subset selection, and synthesis of new attributes) were used in this project. Finally, the constructive induction phase is implemented using an attribute subset selection wrapper for overfitting control and a meta-learning (hyperparameter tuning) system for model selection.

1.2.2. D2K itineraries and data mining process pipeline. Figure 3 illustrates a visual specification and data flow model, called an *itinerary* (Auvil et al., 1999), for the KDD steps in the underwriting project. The itinerary design is documented in the rest of this paper, and the KDD operations in particular are documented in Section 3.

2. Problem

This section presents an overview of the commercial project, sponsored by Allstate Insurance Company, which we will refer to as the One Company project through the rest of the paper. We first describe the decision support objectives of the One Company project and

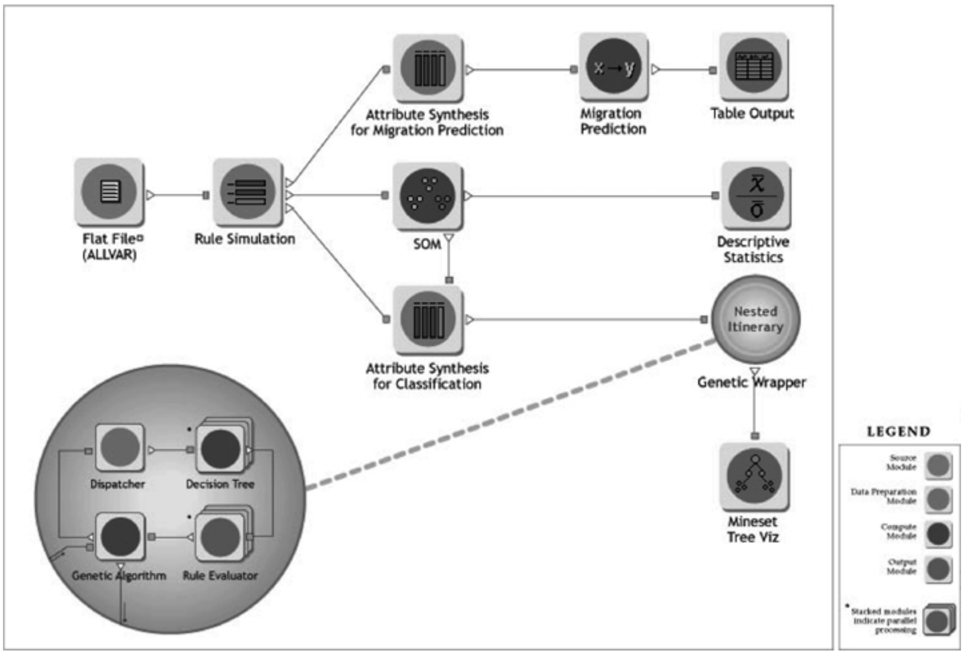

Figure 3. *D2K* itinerary for the Allstate One Company project.

the purpose of KDD, then discuss project-specific issues surrounding the preprocessing of raw data and its transformation into a usable form for KDD.

2.1. Allstate One Company project

The One Company project is an initiative of the Allstate Insurance Company's underwriting division. Its purpose is to reorganize its existing models for pricing of automobile insurance policies in an experimental market, to reduce attrition (loss of customers to competitors) and to control distribution of loss (paid losses due to damage or personal injury claims) across pricing categories. The pricing model is primarily based on a classification problem: given policy descriptors (demographic data about the policyholders, automobile-specific data, driver accident and vehicle code violation history, etc.) identify the *tier* (pricing category) to which a policy belongs. The Allstate underwriting division uses classification to achieve two objectives:

- Develop a predictive model of churn (inter-tier migration) to reduce attrition
- Develop a predictive model of loss and use it to *refine and formulate rules for decision support* in organizing tiers

The focus of this paper is the second objective, but a concurrent phase of the One Company project addressed the first objective and used the same raw and intermediate data sets (with different attributes being selected and synthesized for each).

During initial consultation with the users—decision-makers at Allstate's underwriting division—we established these important characteristics and conjectures regarding the problem:

1. *Fact*: Allstate currently uses a rule-based system to categorize policies. This system was developed approximately 20 years ago using both knowledge elicited from subject matter experts and basic techniques such as linear regression, and has been revised by Allstate's research center to account for concept drift (primarily demographic drift).
   Conjecture: Because compilation of the original rule set was based in part on elicitation and in part on statistical modeling, it is possible to define a KDD problem on recent collections of this historical data.
2. *Fact*: While the rule-based system is the canonical method for placing new customers into tiers and migrating them across tiers (based on updated demographics and accident histories), it does not produce classes that predict loss accurately. Tiers are dissimilar according to analysis of variance (see Section 3.1.1), but predictive accuracy is not significantly better than random for single policies.
   Conjecture: Prediction accuracy might be improved upon by using modern learning methods, such as decision tree induction, on the data originally used for rule extraction, but the prediction target is not necessarily the *paid loss* for a single policy.
3. *Fact*: The entities available for online analytical processing (OLAP) and KDD are expressed in a data set called *ALLVAR* produced by the Allstate research center and used by the underwriting division. This data set contains 471 raw attributes collected by the company from forms submitted by insurance agents and customers and from private records (e.g., payment histories transmitted from the previous insurance company) and public records (e.g., accident and driving violation histories).
   Conjecture: Several data integrity problems affect *ALLVAR*, particularly ambiguities in the definitional data model. There are many irrelevant attributes for predicting targets that represent either of the decision support objectives (churn and loss).
4. *Fact*: The Allstate research center constructs decision trees using *ALLVAR* for evaluation and calibration of the rule-based system. In these experiments, policies are aggregated, and the prediction target is *aggregate loss ratio* (Jonske, 1999).
   Conjecture: Aggregation of policies from the data set may be appropriate for control of model size and attribute selection (both to reduce overfitting), and to make the learning problem more computationally feasible. *Aggregate loss ratio is an appropriate learning target for the loss minimization objective.*
5. *Fact*: The subset of attributes from *ALLVAR* that can be used to make pricing decisions is constrained by current laws (e.g., exact age is prohibited even though exact age categories such as "21–24" are used) and is revised based on continuing legislative decisions. Some attributes in *ALLVAR* are accounted for in other knowledge-based models developed by the Allstate underwriting division and do not belong in experiments involving learning from data.
   Conjecture: Subject matter expertise can be used here to reduce (select and constrain) and synthesize attributes.

The existence and role of the rule-based system established the One Company problem as one of knowledge-based decision support, but the problem definition remained incomplete. This definition was deferred to the descriptive statistics phase, in which the conjectures regarding appropriateness of decision trees, data integrity, and irrelevance were borne out. Related work at Allstate's research center and our own exploratory experiments (using data clustering, descriptive statistics, and pre- and post-visualizations) suggested the data cleaning techniques and emphasis on attribute subset selection that were adopted.

Sources of data and knowledge for the One Company project comprised:

- The *ALLVAR* data set and data dictionary (a rudimentary ontology and data model)
- Subject matter expertise related in knowledge elicitation and engineering phases by the Allstate underwriting personnel, the Allstate research center, and the authors
- KDD experiments using intensively preprocessed versions of *ALLVAR*

2.2. General issues

The One Company project consisted of three main phases: data verification, exploration, and preprocessing; model development; and model refinement. These phases generated the following project milestones:

1. *Phase I*: clean up *ALLVAR* for data mining; pre-select and pre-synthesize attributes
2. *Phase II*: find correct granularity level, representation for training examples
3. *Phase II*: generate decision trees, rules; generate and collect feedback with users of rule-based system; interactively select and synthesize attributes
4. *Phase III*: develop an efficient, flexible wrapper to find relevant attributes in *ALLVAR*
5. *Phase III*: develop a model of loss to aid in evaluation and distribution of tiers

Note that constructive induction (data model-driven, knowledge-driven, and data driven) is applied in each phase to transform the input specification for supervised learning.

2.2.1. Preprocessing: Data cleaning and aggregation. Our data verification efforts focused on unifying standard units for dates and encoding standards for discretized fields (e.g., *class codes* that captured multivariate demographic data such as "single male, 25–29"). Data exploration focused on collection of descriptive statistics from simulated classes (obtained using the rule-based classification system) and discovered classes (obtained using self-organizing maps for dimensionality reduction). Visualization techniques and relevance determination were applied to an early, preprocessed version of *ALLVAR* and submitted to the user. The purpose of these experiments was not only to assess feasibility of constructing models from *ALLVAR* but as a prefilter for irrelevant and redundant attributes, a guide to the data cleaning process (which went through 3 iterations), and a method for establishing the baseline performance of the rule-based system.

Data preprocessing, the most computationally intensive step of the project, applied the rule-based system as a specification for transforming raw *ALLVAR* data into a reduced and synthesized database suitable for direct KDD. It also led to discovery of additional data integrity issues (e.g., ambiguity in normalization factors such as the *number of exposures*) that were resolved through interactive elicitation sessions with the subject matter experts.

Finally, it applied the data dictionary to pre-filter redundant attributes. This first stage of constructive induction was driven by prior knowledge represented as a data model for a very large database (over 1 million records and 471 raw attributes).

Finally, data aggregation used arithmetic methods (summing, averaging, sparse coding and counting) to combine hundreds or thousands of policy records into single training examples. This step was driven by domain expertise (existing practices at Allstate) and preliminary experiments using unaggregated data on a small, but representative, test market (1 line of business out of 8 lines, in 4 states). These stages are represented as *input* and *data preparation* modules in the itinerary shown in figure 3.

2.2.2. Role of relevance determination in decision support. Relevance determination continued to serve a critical role in the direct KDD phases. Data modeling limitations (specifically, the lack of prior relevance knowledge other than that captured in the rule base during the data cleaning phase) necessarily forced most of the attribute synthesis computations to take place early (before aggregation), even though this demanded much greater computational work. Aggregation freed us to apply more computationally intensive methods (such as the genetic wrapper) for attribute subset selection. This design choice was driven largely by user requirements and by the stepwise refinement methodology we selected for constructive induction. The wrapper is depicted as a *learning* module in the itinerary shown in figure 3.

2.2.3. Computational considerations. The primary computational bottleneck, aside from data cleaning and preprocessing of *ALLVAR*, was a "meta-wrapping" technique that we developed for overfitting control in the One Company project. This design choice was motivated by our goal of greater autonomy in model selection for KDD, and constrained by architectural limitations in the high-level wrapper and the computational platform (the Linux and Irix clusters used). These scalability issues are coupled in the *D2K* research program, and are of great interest because of the increase in accessibility to users that greater autonomy and portability would provide.

3. Methodology

This section describes the design of the data mining itinerary shown in figure 3 and the implementation of the data preparation, constructive induction, supervised learning, and visualization modules in this itinerary.

3.1. Exploratory experiments

Exploratory experiments in the One Company project consisted of two categories: data clustering and descriptive statistics on tiers formed using clustering and using the rule-based system; and data characterization and visualization. Both types of experiments were interleaved with the data verification and cleaning steps listed in Section 2.2. The first group occurred during Phase I (development of a data model and data cleaning methodology) and used an early version of *ALLVAR*; the second occurred during Phase II (refinement of the supervised learning problem) and used the final preprocessed version of *ALLVAR*.

3.1.1. Data clustering and descriptive statistics. The objectives of the first type of exploratory experiment, data clustering, were to:

1. Help establish a supervised learning problem
2. Generate a model for comparison with the rule-based system
3. Guide (attribute) quantization and (example) aggregation later in the data flow model

We view cluster definition as a component of *knowledge-guided constructive induction* as proposed in Donoho (1996) and other work. Ideally, this phase should work as a back end to feature construction (synthesis of attributes using techniques such as FOIL (Quinlan, 1990) or genetic programming (Koza, 1992). We found, however, that the role of clustering in the One Company project, was more *descriptive* than *constructive* with respect to our objectives. For the first objective, clustering primarily served to compare the discriminatory potential of *ALLVAR* for two candidate learning targets (paid loss and loss ratio) and to indicate a large number of irrelevant attributes. For the second objective, clustering provided an experimental class definition to compare against the control (rule-based system), again in terms of discriminatory potential. For the third objective, we used scalar quantization methods for sensitivity analysis, to evaluate the feasibility of aggregating many examples.

We applied data clustering methods to *ALLVAR-1*, our first preprocessed version of *ALLVAR*, which was obtained by applying the preprocessing front end of the rule-based system. *ALLVAR-1* contains a total of 254917 records (each a training example for unsupervised learning, with 209 attributes). Because the existing OLAP codes for *ALLVAR* were legacy codes and therefore highly infeasible to port, a complete re-implementation in Java was developed through intensive consultation with the Allstate underwriting division. This part of Phase I required about 25% of the overall project development time and represented about 10% of the overall effort. Clustering methods used on *ALLVAR-1* included Kohonen's self-organizing feature map (Kohonen, 1990) (implemented in *SOM-PAK* (Kohonen et al., 1996)). Preprocessing for SOM consisted of normalization and filtering of sentinel values (intermediate remnants of processing in *ALLVAR-1*), which were replaced with explicit "unknown value" indicators. These two steps each accounted for 3 degrees of magnitude difference in quantization error, which indicates the sensitivity of this implementation of SOM to data impurity (of which we encountered: spurious sentinels, conventions on data delimiters that affect field alignment, and normalization). As in some conventions for simple (naïve) Bayesian inference (Kohavi et al., 1997) and by contrast with unsupervised Bayesian learning methods such as *AutoClass* (Cheeseman et al., 1988) that use expectation-maximization (EM) (Dempster et al., 1977), SOM simply omits missing values from its distance metric computations (Kohonen, 1990; Haykin, 1999). Different runs of SOM (ranging from 10-by-10 to 20-by-20 maps, the latter being reported here) discovered between 6 and 11 clusters in the data, compared with 7 tiers generated by the rule-based system.

Descriptive statistics were collected using the clusters produced by SOM. This required another 10% of the development time and effort. We used these statistics to answer the following queries about the discriminatory capability of *ALLVAR-1* and compare it to that of the rule based system:

1. *Q*: Is there significant dissimilarity between clusters as discovered by SOM? Between tiers as identified by the rule-based classification system?

Table 1. Difference between means and 95% confidence interval, one-way ANOVA (general linear model, Scheffe's test) for pure premium, classes (cluster) from SOM for *ALLVAR-1*.

	1	2	3	4	5	6	7	8	9	10	11
1	–										
2	36.5 ± 28.0	–									
3	−64.6 ± 25.9	−100.1 ± 27.5	–								
4	−157.8 ± 26.8	−194.3 ± 28.4	−94.2 ± 26.3	–							
5	−104.0 ± 24.3	−140.5 ± 26.1	−40.4 ± 23.8	53.8 ± 24.8	–						
6	80.3 ± 24.5	43.8 ± 26.3	143.9 ± 24.0	238.1 ± 25.0	184.3 ± 22.3	–					
7	32.7 ± 22.6	−3.8 ± 24.4	96.3 ± 22.0	190.5 ± 23.1	136.7 ± 20.2	−47.6 ± 20.4	–				
8	164.3 ± 22.9	127.8 ± 24.8	227.9 ± 22.4	322.1 ± 23.5	268.3 ± 20.6	84.0 ± 20.8	131.6 ± 18.5	–			
9	125.7 ± 23.6	89.2 ± 25.6	189.3 ± 23.1	283.5 ± 24.1	229.7 ± 21.3	45.4 ± 21.6	93.0 ± 19.3	−38.6 ± 19.7	–		
10	−157.3 ± 23.0	−193.8 ± 24.8	−93.7 ± 22.4	0.5 ± 23.5	−53.3 ± 20.6	−237.6 ± 20.9	190.0 ± 18.5	−321.6 ± 19.0	−283.0 ± 19.8	–	
11	−310.7 ± 29.9	−347.2 ± 31.4	−247.2 ± 29.5	−152.9 ± 30.3	−206.7 ± 28.2	−391.0 ± 28.4	−343.4 ± 26.7	−475.0 ± 27.0	−463.4 ± 27.6	−153.4 ± 27.0	–

Table 2. Same ANOVA results for pure premium, classes (tiers) from rule simulations on *ALLVAR-2*.

	A	B	C	D	E	G	H
A	–						
B	46.6 ± 85.2	–					
C	217.3 ± 249.6	170.7 ± 252.1	–				
D	168.9 ± 67.8	117.2 ± 76.3	−53.5 ± 246.7	–			
E	−101.6 ± 516.88	−148.2 ± 518.1	−318.9 ± 568.7	−265.4 ± 515.5	–		
G	321.5 ± 114.7	274.8 ± 120.0	104.1 ± 263.5	157.6 ± 108.2	423.0 ± 523.7	–	
H	270.6 ± 226.7	224.0 ± 229.4	53.3 ± 328.2	106.8 ± 223.5	372.2 ± 559.1	−50.8 ± 241.9	–

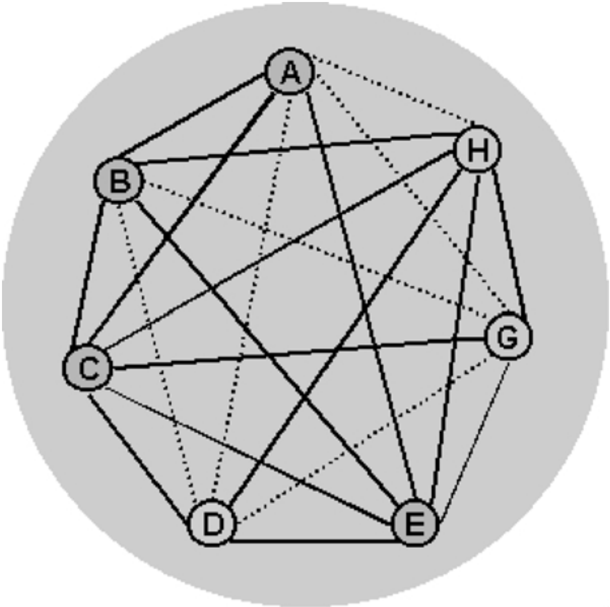

Figure 4. ANOVA similarity graph for tiers from rule simulations, *ALLVAR-2* (dotted lines: Significant differences between tiers, at 95% level of confidence).

Descriptive statistics: analysis of variance (ANOVA), which produced the output shown in Tables 1 and 2 and figure 4.

2. *Q*: What is the distribution of premiums within clusters as discovered by SOM? Within tiers as identified by the rule-based classification system?
   Descriptive statistics: calculation of mean and variance of pure premium by cluster, which produced the output shown in Tables 3 and 4 (the descriptive statistics module in our D2K system produced similar statistics for all 208 attributes of *ALLVAR-1*)

We found it useful to calculate both inter-category descriptive statistics (item 1) and intra-category statistics (item 2). Inter-category statistics allow some comparison between

Table 3. Intra-cluster descriptive statistics for pure premium: SOM clusters (*ALLVAR-1*).

Class (Cluster)	Size	Mean	Stdev
1	17134	542.65	662.84
2	13683	579.13	652.76
3	18517	479.06	538.10
4	16026	384.82	414.91
5	24346	438.65	537.03
6	23316	662.96	659.70
7	36781	575.35	563.56
8	33225	706.92	669.99
9	28228	668.35	643.01
10	32827	385.34	383.80
11	10834	231.91	311.29

Table 4. Intra-cluster descriptive statistics for pure premium: rule simulation tiers (*ALLVAR-2*).

Class (Tier)	Size	Mean	Stdev
A	86469	303.93	4422.35
B	61334	350.84	3594.63
C	4397	520.76	3018.57
D	165736	467.92	4398.46
E	997	200.89	1024.93
G	25683	625.26	7361.07
H	5384	574.35	4055.49

classification methods—here, data-based (SOM, a competitive clustering algorithm based on Euclidean distance (Kohonen, 1990) and knowledge-based. We must be careful, however, to specify a proper cluster definition method, including the cluster *formation*, *segmentation*, and *labeling* algorithms.

1. *Formation*: we used an existing implementation of SOM (Kohonen et al., 1996) and data preparation modules for normalization and integrity checking
2. *Segmentation*: we used a simplification of learning vector quantization (LVQ) (Gersho and Gray, 1992; Haykin, 1999), an instance-based learning (IBL) (Aha et al., 1991; Mitchell, 1997) algorithm that computes the nearest-neighbor regions (Voronoi cells) about the cluster representatives (or "centers").[1]
3. *Labeling*: our IBL technique implicitly defines a labeling algorithm. First, assign integer labels to each cluster representative; second, assign each new data point the label of its nearest neighbor (found by querying the Voronoi diagram).

Inter-category ANOVA (testing the *difference in mean pure premium* among clusters) indicated that there was significant dissimilarity among all of the 11 clusters output by

SOM and our simplified LVQ algorithm (applied to *ALLVAR-1*), as shown in Table 1 and figure 4. In all but two of the paired tests in Table 1, the difference between estimators is significant at the 95% level of confidence, indicating up to 11 equivalence classes of pure premium according to the criterion defined using our simplified SOM/LVQ labeling algorithm. By contrast, the inter-category ANOVA for tiers, output by the rule simulations (applied to *ALLVAR-2*), showed only sporadic dissimilarity.[2] As shown in Table 2, only 6 pairwise tests show a difference that is significant at the 95% level of confidence. This indicates at least 3 equivalence classes (e.g., {{A, B, C, E} {D} {G, H}}; {{A, B, C} {D, E} {G, H}}; etc.), but shows that tiers are not strong discriminators of expected premium.

Intra-category statistics also helped us compare the conditional distribution of a candidate learning target (loss) across categories. The cluster means and variances listed in Table 3 are from the data clusters produced by our SOM-based cluster definition (formation, segmentation, and labeling) algorithm—the same output that generated Table 1. It was this result that led us to consider aggregation methods (subsequently corroborated by the Allstate underwriting division and research center) and to focus on loss ratio (the aggregate quantity to be equalized in the performance element) as a classification target.

Our final quantization step was scalar quantization of attribute values in *aggregated* data points using the binning methods in SGI *MineSet*. This is described together with our aggregation technique in Section 3.2.2.

3.1.2. Data characterization and visualization. Our exploratory experiments continued using *ALLVAR-1* and *ALLVAR-2* (a “cleaned” data set described in Section 3.2.1 below). Our machine learning experiments in Phase II required about 15% of development time and 20% of the overall effort, and the visualization and data characterization steps accounted for about half of this. These were carried out using *MLC++* and *MineSet*, primarily using the *column importance* (relevance determination using a *filter* (Kohavi, 1998; Kohavi and John, 1997) based on a cross-entropy score), and the *Evidence Visualizer*. A summary of output from column importance on *ALLVAR-2* is shown in Table 5. Many other experiments were conducted on *ALLVAR-2* prior to aggregation, but these are omitted for brevity due to our subsequent choice to use aggregation for all *MLC++* and *D2K* batch experiments.

The use of *MineSet Tree Visualizer* on a variety of decision trees produced using different exploratory aggregation and quantization methods (uniform-population binning, uniform-width binning, and scalar quantization by hand) led us to decide on uniform aggregation and 1000 policy records as the appropriate aggregation granularity. The motivation for this was the severe overfitting and qualitative tree complexity observed in the *Tree Visualizer* using smaller aggregates, especially one example per policy record.

3.2. Data cleaning

Our data cleaning stage consisted not only of preprocessing, integrity checks, and normalization, but also aggregation, quantization, and histogramming. We have described (in Sections 2.2.1 and 3.1.1) some of the elementary but consequential integrity issues—

Table 5. Attributes selected by applying *MineSet* column importance to *ALLVAR-2*.

Rank	Name	Cumulative score	Meaning
1	numloss	85.59	Number of losses (in last 5 years) incurred on this policy
2	leastlic	75.45	Years most recent driver on policy has had license
3	youngage	75.45	Age of youngest driver on policy
4	tier	75.45	Classification assigned by rule based system (*ALLVAR-2*)
5	drivers	75.4	Drivers insured on this policy
6	cars	75.38	Vehicles insured on this policy
7	yda	75.35	Internal historical code
8	oldage	75.35	Age of oldest driver on policy
9	classcd1	75.33	Composite attribute: sex, age, marital status
10	mostlic	75.32	Years most experienced driver on policy has had license
11	classcd6	75.32	Composite attribute: driver-specific discounts
12	classcd7	75.32	Composite attribute: vehicle-specific discounts
13	prabdin5	75.31	Number of bodily injury, collision claims in last 5 years
14	o1lic5	75.3	Has the primary operator been licensed 5 years?
15	o1lic3	75.3	Has the primary operator been licensed 3 years?
16	classcd9	75.3	Composite attribute: geographically-specific discounts
17	kids	75.29	Number of children in household of policy holders
18	classcd2	75.29	Composite attribute: annual mileage
19	oldtier	75.28	Classification assigned by rule based system (*ALLVAR-1*)
20	classcd3	75.28	Composite attribute: usage (commuting, pleasure, etc.)
21	o1lic1	75.28	Has the primary operator been licensed 1 year?
22	add_drop	75.28	Difference between add, drop dates of policy
23	paycode	75.28	Payment status to current insurer (outstanding debts, etc.)
24	est_miles	75.28	Estimated mileage on odometer of primary vehicle
25	yrcurjob	75.27	Years primary policy holder has been at current job
26	mttable	75.27	Merit table (composite, internal predictor variable)
27	pricode	75.27	Historical field: prior insurance code
28	limded	75.27	Limited deductible
29	accsurch	75.27	Accumulated surcharge
30	term_eff	75.27	Difference between termination, effective dates of policy

including normalization and sentinel values—that arose during computation of descriptive statistics. In developing our data model, we reviewed the data dictionary carefully to verify units, bounds, and ontological consistency (especially for boundary cases) of attributes. This led to a rudimentary abstract data type (ADT) definition and uncovered two data cleaning issues: the representation of dates (we found 3 inconsistent formats to resolve) and the definition of "exposures" (a measure of a customer's coverage over time that was important in both the classification and migration problems).

3.2.1. Preprocessing: Rule simulations, attribute synthesis, and column importance. The ADT developed from the data dictionary generated two specifications: first, the simple type definitions (integer-valued, continuous-valued, and enumerative attributes) used by *MLC*++ and *MineSet*; second, a requirements specification for *ALLVAR-2*. A second data preprocessing code was developed in Java using this specification of requirements. The development time was approximately 20% of the project total, the effort expended, over 15% of the project total. We also estimate that over half of all the computational resources expended in the One Company project were devoted to simulating the revised rule base, to transform data from the original representation to that of the newly synthesized attributes.

ALLVAR contains 471 attributes that are reduced to 86 by these rule simulations, to produce *ALLVAR-2*. This process eliminated, using a trivial amount of computation, attributes previously known to be irrelevant and those that were handled by components of Allstate's decision support system other than our performance element (e.g., geographic attributes such as zip code). Through interactive elicitation, we also eliminated those that were beyond the scope of our KDD problem even if relevant (e.g., insurance agent identity).

In addition, our preprocessing (data preparation) modules synthesized new attributes based on combined domain expertise from the Allstate underwriting group and exploratory experiments. For example, subset regression in *SAS* (using the *MAX-R* algorithm) frequently resulted in the simultaneous selection of attributes that denoted the starting and ending dates of intervals (e.g., a customer's add and drop date, a policy's termination and effective date). These were *not* selected using single-variable regression or *MineSet Column Importance* (a mutual information, i.e., cross-entropy, criterion), but simply synthesizing *interval duration* attributes resulted in their inclusion by both of these greedy algorithms. Finally, attribute synthesis also applies known formulae and algorithms to condense many attributes (e.g., those describing *coverages*) into single historical attributes (e.g., *exposure* counts). Application of these functional definitions to 2.7 million records of *ALLVAR* (across 2 lines of business, in 4 states) was by far the most computationally intensive step of the One Company project.

Table 5 shows the results of applying *MineSet Column Importance* to a sample of 350000 records from *ALLVAR-2*. This was the largest subset that could be handled due to memory limitations (and was the determinant of our sample size in subsequent experiments, for which we refer to this sample as "*ALLVAR-2*"). Note that the increments in cumulative score do not decrease monotonically, as the criterion is greedy. Note also that the top 30 attributes selected include some that behave as decision lists (e.g., o1lic5, o1lic3, o1lic1), some correlated subsets (e.g., leastlic, youngage, and o1lic1), and some *composite attributes* used in Allstate's corporate database systems. An example of a composite attribute and its semantic content are shown in Table 6. We discuss the aggregation of examples by the 30 attributes of Table 5, and the further selection of these attributes, in the rest of Section 3.

3.2.2. Aggregation. Aggregation, quantization, and histogramming methodology for *ALLVAR-2* was developed after exploratory experiments, attempting to classify policy records individually, resulted in severe overfitting. These produced decision trees with 2000–6000 nodes for 20000–100000 training examples. Subsequent consultation with the Allstate research center confirmed that aggregation was a standard practice for the KDD

Table 6. A composite attribute and the aggregation method (histogrammed 1-of-C coding) used.

Code	Frequency	Meaning	1-of-C coding
1	230569	Adult	1 0 0 0 0 0 0 0 0
2	61945	Retired adult	0 1 0 0 0 0 0 0 0
3	21638	21–49, single female	0 0 1 0 0 0 0 0 0
4	5120	25–49, single male	0 0 0 1 0 0 0 0 0
5	8698	21–24, male	0 0 0 0 1 0 0 0 0
6	4671	20 and under, male	0 0 0 0 0 1 0 0 0
7	12643	30–49, single male	0 0 0 0 0 0 1 0 0
8	4715	20 and under, female	0 0 0 0 0 0 0 1 0
9	1	Unknown (spurious)	0 0 0 0 0 0 0 0 1

experiments used in their decision support operations (such as rule refinement). This section documents and justifies the methodology used.

Table 6 shows the denotation of a *composite*, nominal attribute called classcd1 (Class Code 1), which encodes several well-known demographic categories that are hypothesized (and have been demonstrated in some studies (Porter, 1998) to be indicators of automobile insurance risk. Note that these codes are exhaustive but *not* mutually exclusive; among other reasons, this is because multiple drivers can be covered per vehicle and per policy (e.g., the attribute value "Adult" subsumes most multi-driver records). Generally, for composite attributes, the most specific applicable category value is recorded. We use a 1-of-C encoding (Sarle, 1999) to encode composite attributes, simply because it allows us to produce a histogram easily when aggregating new attributes. This results in 27 "dense" attributes being coded as 97 "sparse" ones.

Table 7 summarizes the entire aggregation specification for the 30 most *individually* relevant attributes found using a filter (*MineSet Column Importance*) as a "quick rejection" test. Note that the aggregation method depends on the context (the denotation of the attribute in the data model) and not merely on the attribute type (including the units). For example, binary attributes are summed to obtain totals. For variable-sized aggregates, such as stratified samples, these must be divided by the aggregate size to obtain frequencies. We use constant-sized aggregates for *ALLVAR-2*, so this distinction is obviated, as is that between sums and average. It is critical, however, to note the difference in user interpretation that results even if unweighted (identical-sized) samples are used to produce one example each using the above specification. For example, "cars" and "kids" are reported as averages for predicting a target that is a *ratio* of functions of sample size, while the number of operators licensed for less than one year is a sum.

Our overall methodology for relevance determination incorporates descriptive statistics, data cleaning, column importance, aggregation and histogramming, and attribute subset selection (an interleaved select-combine-select strategy) to "funnel" the 471 attributes into 23 as shown in figure 5. Three additional attributes (all internal fields used by other components of the Allstate underwriting decision support system) were eliminated at the column importance stage.

Table 7. Aggregation specification for *ALLVAR-2* data model.

Name	Original type	Units	Aggregation method	New attributes
numloss	Integer	Count	Average	1
leastlic	Integer	Years	Average	1
youngage	Integer	Years	Average	1
tier	Nominal	N/A	1-of-C coding, histogramming	7
drivers	Integer	Count	Average	1
cars	Integer	Count	Average	1
yda	Integer	Years	NONE	0
oldage	Integer	Years	Average	1
classcd1	Nominal	N/A	1-of-C coding, histogramming	9
mostlic	Integer	Years	Average	1
classcd6	Nominal	N/A	1-of-C coding, histogramming	9
classcd7	Nominal	N/A	1-of-C coding, histogramming	10
prabdin5	Binary	N/A	Sum	1
o1lic5	Binary	N/A	Sum	1
o1lic3	Binary	N/A	Sum	1
classcd9	Nominal	N/A	1-of-C coding, histogramming	11
kids	Integer	Count	Average	1
classcd2	Nominal	N/A	1-of-C coding, histogramming	9
tier2	Nominal	N/A	NONE	0
classcd3	Nominal	N/A	1-of-C coding, histogramming	10
o1lic1	Binary	N/A	Sum	1
add_drop	Integer	Days	Average	1
paycode	Nominal	N/A	1-of-C coding, histogramming	6
est_miles	Integer	Miles	Average	1
yrcurjob	Integer	Years	Average	1
mttable	Nominal	N/A	NONE	0
pricode	Nominal	N/A	1-of-C coding, histogramming	8
limded	Integer	Count	Average	1
accsurch	Integer	Count	Sum	1
term_eff	Integer	Days	Average	1

3.3. Scalable supervised learning for large databases

One important innovation of this *D2K* system is the automation of our attribute subset selection system using a genetic algorithm-based performance-tuning wrapper. We view relative weights for validation set accuracy, input complexity (attributes selected), and model complexity (tree size) as *hyperparameters* to be optimized over as well (Neal, 1996). The purpose of reducing the attribute set is to increase the comprehensibility of the model (in this case,

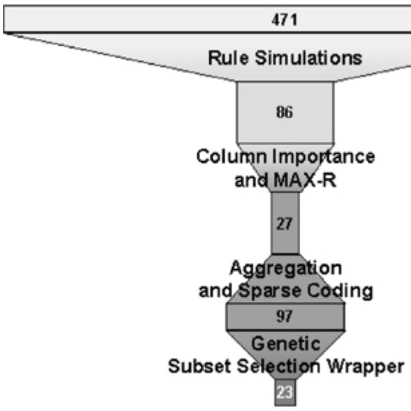

Figure 5. Change of representation process.

the decision tree and resultant rules) through overfitting avoidance. Our choice of model is driven by the inherent scalability of the model (the ability to construct decision trees that grow in complexity based on the number of attributes) and lack thereof (sensitivity of these trees to overfitting due to irrelvant and redundant attributes *observed over many training examples*. We find that combining a wrapped decision tree inducer (Kohavi and Sommerfield, 1996) and aggregation produces a highly extendible classification learning system for data sets such as *ALLVAR*. This approach makes decision tree learning more robust to mixed numerical and symbolic data, many irrelevant attributes, a data model with fields at differing granularity, and medium to large data volume (hundreds of thousands to millions of records). Our goal in developing intelligent data mining agents is to produce more configurable yet more autonomous wrappers and more transparency to the user. As we shall discuss in Section 3.4, however, this autonomy is purchased at the cost of intensive computation.

The *simple inducer* phase, comprising experiments using *MineSet* and *MLC++* without attribute subset selection filters (Kira and Rendell, 1992; Kononenko, 1994) and wrappers (Kohavi and John, 1997), accounts for 15% of development time and 20% of overall effort.

3.3.1. Decision tree induction. Initial results using *ID3* as applied to aggregated *ALLVAR-2* (97 attributes) are shown in Table 8. Test set accuracy is significantly better than random (uniform prediction over the classes, or bin labels) in all cases except with 4 target bins. This is clearly a very weak criterion, as the utility of prediction accuracy far under 50%, even as a coarse-grained recommender system, is negligible. The Allstate underwriting team indicated, however, that even the weak 2-bin predictor can be useful in generating *explanations* for high and low expected loss ratio. This is because purity tends to be high at low depths of the tree (indicating a potential for pruning). Overfitting control is indeed possible with reduced error pruning, even for unaggregated data, but has limited effect because of relatively irrelevant and correlated attributes. Two things are needed to truly improve generalization quality: aggregation, and an attribute subset evaluation function that controls the balance between training set accuracy and generalization criteria (model size, model input size). Aggregation improves generalization in decision trees for *ALLVAR-2*:

Table 8. ID3 decision tree performance using aggregated *ALLVAR-2*.

Target bins	Average tree size	Test set accuracy
2	24	61.54 ± 4.52
3	36	43.59 ± 4.60
4	39	28.21 ± 4.18
5	36	39.71 ± 5.85
6	50	23.93 ± 3.96
7	52	25.64 ± 4.05
8	53	15.38 ± 3.35

none of the many exploratory decision trees we constructed using unaggregated *ALLVAR-2* had test set accuracy significantly better than random. A constructive induction wrapper can also improve generalization in decision trees if its objective is to reduce overfitting.

3.3.2. Visualization and interpretation. Visualizations generated during development of the *D2K* One Company system included:

1. The U-matrix plot (Kohonen map visualization) on *ALLVAR-1*, using *SOM-PAK* (Kohonen et al., 1996)
2. The ANOVA similarity graph of figure 4 on *ALLVAR-2*, using *SAS*
3. The cross-entropy scores from *MineSet Column Importance* for the 30 attributes of *ALLVAR-2* described in Table 5, using *MineSet Evidence Visualizer* (*EviViz*) (Kohavi, 1998)
4. About 20 sets of decision trees using *ALLVAR-1* and *ALLVAR-2*, using *MineSet Tree Visualizer* (*TreeViz*) (Kohavi, 1998), a 3-D decision tree visualization and navigation system, and *AT&T GraphViz* (Krishnamurthy, 1995), a 2-D display package for which *MLC++* generates output.

These visualizations were conducted during the first 9 months of the 10-month project, and delivered to the user on a biweekly to weekly basis.

3.4. Meta-learning: Adaptive wrappers

We use a constructive induction wrapper to control three factors:

1. Overfitting as detected through validation set accuracy
2. Tree size
3. Selected attribute subset size

Kohavi et al. have conducted extensive research in formulating attribute subset selection as a state space search (Kohavi, 1995; Kohavi and John, 1997). In recent experiments by Cherkauer and Shavlik (1996), Raymer et al. (1997), and Dejong et al. (1993), the efficiency of using genetic algorithms as wrappers for performance tuning in supervised inductive learning and has been demonstrated. We observed, however, that more *autonomy* is needed in allowing the user to indirectly specify the balance among the three factors above. That is, the weights given to these factors in the fitness evaluation function of the genetic wrapper should themselves be hyperparameters. This is a topic of continuing research; in this paper, we outline a procedure for specifying these hyperparameters and our findings using a genetic wrapper implemented in *D2K*. This wrapper accounts for 10% of amortized development time (over projects in which it has been used—2 at the present time) and 10% of the overall effort.

3.4.1. Tunable attribute subset selection. The frequency of irrelevant attributes, and especially correlated attributes, is high, as described in Section 3.2. Figure 5 illustrates this "column-wise" sparseness of information. Some shortcomings of single-variable and subset regression (*MAX-R*) and score-based ranking (*Column Importance*) can also be seen in retrospect. Namely, they act as attribute *filters* (which evaluate attributes, singly or in tuples, according to a criterion external to the inducer used) rather than wrappers (which treat the input specification of the supervised learning problem as a trainable hyperparameter, and optimize this in an "outer loop") (Kohavi, 1995; Kohavi and John, 1997). We hypothesized that a simple wrapper for attribute subset selection would improve performance, while a full constructive induction wrapper (as implemented using the entire *D2K* itinerary shown in figure 3) would yield even greater benefits.

3.4.2. D2K itinerary. Figure 6 shows a screen shot depicting the nested sub-itinerary of the *D2K* itinerary shown in figure 3 (the fragment in the circle). A noteworthy property of this design is its autonomy for development of improved (reduced and synthetic) input

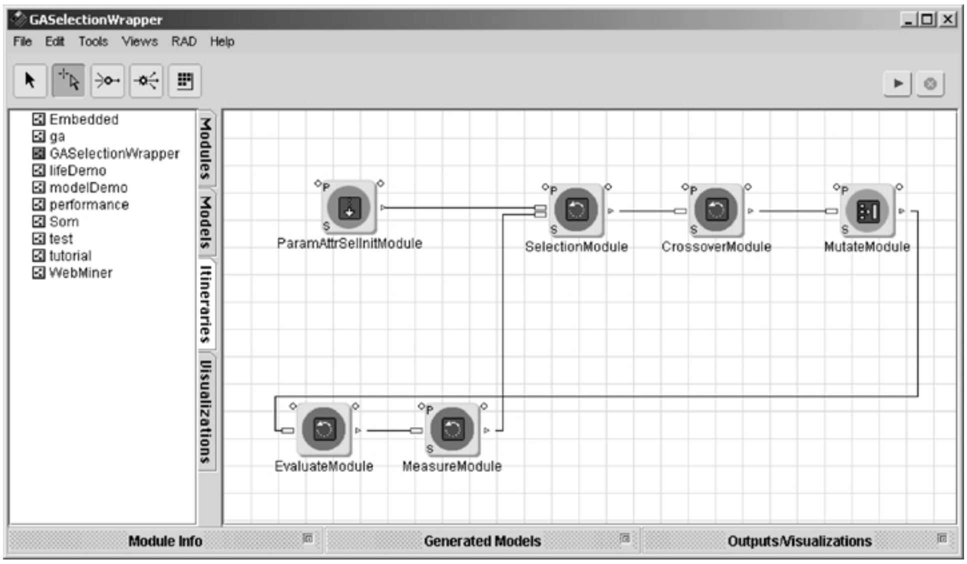

Figure 6. Portion of *Jenesis* nested itinerary within *D2K*.

attributes. We designed *D2K* as an interactive, flexible, and efficient system for adapting data representations to suit decision support objectives of KDD such as the business objectives of the Allstate underwriting division. For example, the *D2K* master itinerary contains reused modules (imported, stand-alone Perl and high-performance Java modules for data preparation); newly constructed modules (the GA of figure 6); integrated research and commercial codes in offline mode (*SAS, MLC++*, and *MineSet* via batch mode scripts); and a new communications mechanism (*Beowulf*).

3.4.3. Genetic algorithm (Jenesis). A GA is ideal for implementing a constructive induction wrapper, attribute subset selection in particular, and a tunable CI wrapper as well (even as an inner loop only). We used a reimplementation in Java of the *Genesis* package of Grefenstette (1990), which we called *Jenesis*. We note that the *Jenesis* wrapper is an advance over the work of Raymer et al. (1997) and Cherkauer and Shavlik (1996) in that it adjusts "empirically determined" constants dynamically rather than assuming that a single optimum exists for a large set of KDD problems. This is preferable to empirically calibrating hyperparameters as if a single "best mixture" existed. Even if a very large and representative corpus of data sets were used for this purpose, there is no reason to believe that there is a single *a posteriori* optimum for hyperparameters such as weight allocation to model size, input complexity, and training set accuracy in the constructive induction wrapper.

The design of the *Jenesis* wrapper illustrated in figure 6 is as follows.

1. The *master controller* for the itinerary runs in a Java virtual machine. We have tested this controller using desktop and portable PCs running *Windows*, *Linux*, *SunOS*, and *MacOS*. The master controller implements *Jenesis* (shown in the screen shot) and manages slaves that concurrently evaluate members of its population (size 100 in our One Company experiment). Each individual is encoded as a bit mask denoting inclusion or exclusion of an attribute (i.e., the parallel search in attribute subset space is conducted using a masking GA (Raymer et al., 1997).
2. 8 slave processes distributed across 2–6 cluster hypernodes (documented in Section 3.4.4) run identical copies of an *MLC++* -based application. Each evaluates the attribute subset it is given by training on a segment of the data (1/3 to 3/5 of all examples) and returns the number of attributes correctly classified in the validation set (a hold-out set, containing 1/5 to 1/3 of the data).
3. The master GA collects the fitness components for all members of its population and then computes the weighted function:

$$a \cdot \frac{c_v}{n_v} + b \cdot \frac{t}{s} + c \cdot \frac{s}{m}$$

   where a, b, and c are coefficients, summing to 1.0, for percent validation set accuracy percent maximum tree size, and percent saturation of the attribute subset. c_v is the number of correctly classified validation cases, n_v is the total number of validation cases, t is the decision tree size in nodes, s is the number of attributes selected in the current specification (the number of 1 bits in the mask string), and m is the total number of attributes. In Cherkauer and Shavlik (1996), $a = 0.75$, $b = 0.125$, and $c = 0.125$.

4. The "inner loop" iterates until termination (300 generations in our One Company experiment; fewer for the other "toy" experiments listed in Section 4). The weights a, b, and c are then validated using the test set (another hold-out set, containing 1/5 to 1/3 of the data) and we select the "best of the outer loop" by inspecting the test set accuracy.

3.4.4. Clusters (NCSA, KSU). We use a *Beowulf* cluster to implement the *Jenesis* wrapper. Beowulfs are commodity-off-the-shelf (COTS) networks of personal computers and single-user workstations running open-source operating system software (typically *Linux*) and communicating using message-passing protocols (typically MPI, PVM, or direct sockets-based communication) (Sterling et al., 1999). Beowulfs have become increasingly popular for compute-intensive applications because they provide a relatively inexpensive way to implement distributed and parallel processing for functionally (task-level) parallel problems. As we describe below, this property holds true for high-performance KDD applications such as performance tuning wrappers.

Table 9 summarizes the technical specifications for the experimental platforms at NCSA and Kansas State University. The primary design rationale for these network-of-workstation clusters is the minimization of price-to-performance ratio, but this measure must also be amortized across the expected platform size. That is, due to the economy of scale in building mid-sized (32 to 128-node at the present time) to large-sized (over 128 nodes) clusters, we must consider:

1. The amortized cost of the cluster (networking, memory, supporting infrastructure—operating system software, commercial KDD and information visualization codes, etc.)
2. The size of a hypernode.
3. The performance gain in using large versus small hypernodes. Our general finding is that scalability improves with larger hypernodes, but there is a narrow economy of scale due to the nonlinear cost of symmetric multiprocessor (SMP) systems.

To illustrate the third point, consider that the One Company *Jenesis* wrapper required just under 1 hour to complete 300 generations ($300 \times 100 = 30000$ decision trees on 350 aggregate training examples with an average of over 25 attributes) on *Valinor*, the larger of our two Beowulf clusters. On *Beoworld*, it required over 4 hours. On a single-processor, 400M Hz Pentium II workstation running *Linux*, it would require well over 30 hours. The current price ratio of a typical 400 MHz Pentium II workstation to *Valinor* (a new system) is 20:1.

Table 9. Technical specifications for KDD clusters.

Cluster	Nodes	Processors	Network type	Processor type/speed	Configuration	Memory per processor
Beoworld (NCSA)	6	8	100-mbps Ethernet	400 MHz Pentium II	2 2-way SMP 4 uniprocessor	128 Mb RAM, 512 K cache
Valinor (Kansas state)	2	8	Gigabit Ethernet	500 MHz Pentium III-Xeon	2 4-way SMP	1 Gb RAM, 1 Mb cache

Table 10. Results from *Jenesis* for One Company (5-way cross validation), representative data sets.

	ID3		*FSS-ID3* wrapper		*Jenesis* wrapper	
Data set	Test set accuracy (%)	Attributes selected	Test set accuracy (%)	Attributes selected	Test set accuracy (%)	Attributes selected
ALLVAR-2, 5 bins	39.71	36.40 ± 1.96	44.00	10.60 ± 4.32	44.86	20.8 ± 1.47
Mushroom	99.82	6/22	99.89	5/22	99.89	5/22
Iris	94.00	4/4	98.00	1/4	98.00	1/4

We observed a linear speedup with the use of multiple processors, but this is improved further through efficient I/O management. Each slave process receives a bit string and performs an on-line, in-memory query of a copy of the data set that is loaded in once per hypernode and memory-mapped. This provides a speedup over time (due to the amortized cost benefit) and as a linear function of the number of processors per hypernode (1 or 2 in Beoworld, 4 in Valinor).

4. Results

4.1. Performance

Table 10 summarizes the performance of the *ID3* decision tree induction algorithm (Quinlan, 1985) and the state-space search-based feature subset selection (FSS) wrapper in *MLC++* (Kohavi and Sommerfield, 1996) compared to that of *Jenesis*. We used a version of *ALLVAR-2* with 5 bins of loss ratio. Wall clock time for the *Jenesis* and *FSS-ID3* wrappers was comparable. As the table shows, both the *Jenesis* wrapper and the *MLC++* wrapper (using *ID3* as the wrapped inducer) produce significant improvements over unwrapped *ID3* in classification accuracy and very large reductions in the number of attributes used. The test set accuracy, and the number of selected attributes, are averaged over 5 cross validation folds (70 aggregate test cases each). Results for 2 data sets from the Irvine database repository that are known to contain irrelevant attributes are also positive.

Table 11 presents more descriptive statistics on the 5-way cross-validated performance of ID3, FSS-ID3 (the *MLC++* implementation of *ID3* with its feature subset selection wrapper), and *Jenesis*. Severe overfitting is quite evident for *ID3*, based on the difference between training and test set error (perfect purity is achieved in all 5 folds) and the larger number of attributes actually used compared to the wrappers. *Jenesis* and *FSS-ID3* perform comparably in terms of test set error, though *FSS-ID3* has less difference between training and test set error and *Jenesis* is less likely to overprune the attribute subset. Note that *FSS-ID3* consistently selects the fewest attributes, but still overfits (*Jenesis* achieves lower test set error in 3 of 5 cross validation cases). The test set errors of *Jenesis* and *FSS-ID3* are not significantly different, so generalization quality is not conclusively distinguishable in this case. We note, however, that excessively shrinking the subset indicates a significant tradeoff regarding generalization quality.

		Cross validation segment						
		0	1	2	3	4	Mean	Stdev
Training set accuracy (%)	ID3	*100.0*	*100.0*	*100.0*	*100.0*	*100.0*	*100.0*	0.00
	FSS-ID3	55.00	54.29	67.86	50.36	60.71	57.64	6.08
	Jenesis	65.71	67.14	71.43	71.43	55.71	66.29	5.76
Test set accuracy (%)	ID3	41.43	*42.86*	28.57	41.43	44.29	39.71	5.67
	FSS-ID3	*48.57*	35.71	*34.29*	47.14	54.29	44.00	7.74
	Jenesis	41.43	*42.86*	31.43	*52.86*	*55.71*	***44.86***	8.69
Attributes selected	ID3	35	35	37	40	35	36.40	1.96
	FSS-ID3	7	8	7	13	18	10.60	4.32
	Jenesis	20	19	22	20	23	20.80	1.47

Figure 7 shows the learning curve for *ALLVAR-2* using the *Jenesis* wrapper. We observed total domination by a single individual within 250 to 400 generations, but it is likely, based on the suboptimality of at least some of the subsets found, that niching methods (e.g., sharing (Goldberg, 1989) can preserve necessary diversity among nondominated (Pareto-optimal) subsets. We are incorporating such features into later versions of *Jenesis*. Furthermore, we observed that for any pair of cross-validation folds, the resulting subsets produced by *Jenesis* had only about half of their approximately 20 attributes in common. This instability indicates the need for more systematic validation experiments and possibly for Bayesian integration methods (Neal, 1996), for which the GA and other Markov chain Monte Carlo methods may be suitable.

The classification model was used to audit an existing rule-based classification system over the same instance space, and to calibrate an underwriting model (to guide pricing decisions for policies) for an experimental market.

Table 11. Distribution of resources for One Company project.

Task	Development time	Overall effort (%)	Computational resources (%)
ALLVAR-1 rule simulations	25% (1 FTE)	10	30
Descriptive statistics	10% (2.5 FTE)	10	1
ALLVAR-2 rule simulations	20% (2 FTE)	15	50
MineSet/MLC++ experiments	15% (3 FTE)	20	5
D2K development and experiments	10% amortized (2.5 FTE)	10	10
Collecting and interpreting results	15% (4 FTE)	25	0
Generalization of *D2K* modules	5% amortized (5 FTE)	10	4

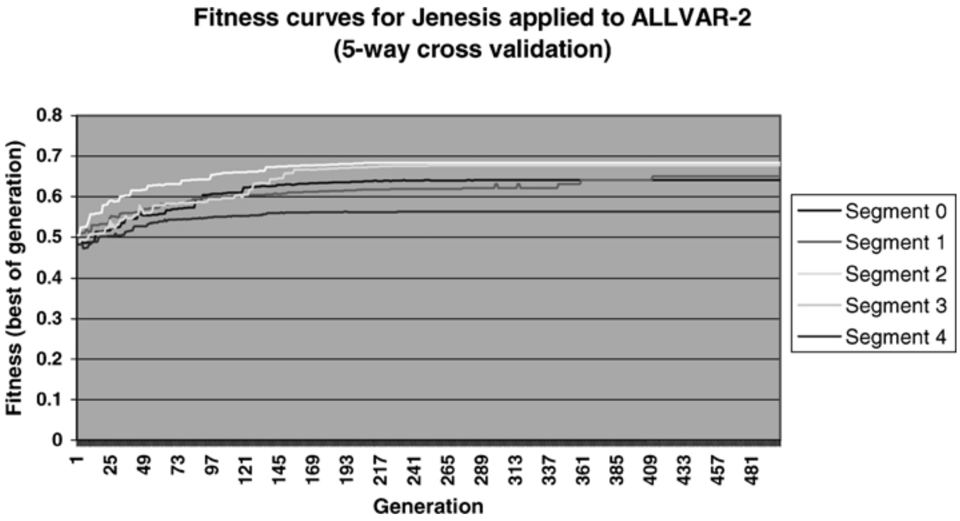

Figure 7. Learning curves (fitness).

4.2. Lessons learned

The Allstate underwriting division's foremost objective was to answer the question "What are the primary determinants of the targets (pure premium, total loss, and loss ratio) among attributes defined over *ALLVAR* fields?" Answering this question was not, however, as straightforward a matter as running *MineSet Column Importance* after determining the target and cleaning up the data model. To successfully produce a simple and comprehensible classification model, we had to consider joint relevance and be able to *explain why* an attribute was important (in rule-based and visual terms). To this end, we developed a generic pipeline for attribute synthesis, reduction (pre-filtering), transformation, and subset selection as shown in figure 5. This pipeline is implemented using the *D2K* itinerary shown in figure 3, and its core novel component is the genetic wrapper shown in figure 6.

We have observed that the aggregation method scales well across lines of business (the indemnity and non-indemnity companies) and states. This was demonstrated using many of our decision tree experiments and visualizations using *ALLVAR-2* samples and subsamples by state.

Our current and future work includes the following extensions and experiments with the One Company itinerary:

1. *Controlling solution* (*subset*) *diversity* using niching and crowding methods (Goldberg, 1989)
2. *Using Bayesian learning methods* to integrate over subsets (Neal, 1996), e.g., combining solutions from the Pareto-optimal front (Goldberg, 1989)

3. *Extension of Jenesis to bias optimization problems* with more degrees of freedom (e.g., continuous weights; full constructive induction including feature construction and extraction (Donoho, 1996)).
4. Comparison with parallel, stochastic (non-GA-based) state space search (Kohavi, 1995; Hsu, 1998; Hsu et al., 2000)
5. Using our data clustering techniques on the prediction problem (inter-tier migration) and to track concept drift
6. Comparing SOM/LVQ to dimensionality reduction methods such as factor analysis (FA), principal components analysis (PCA), and *AutoClass*
7. Adapting incremental clustering methods such as LVQ to take user-specified hyperparameters
8. (Manual, interactive) rule refinement using the SOM output
9. Automating validation experiments in *D2K*
10. Experimentation with different histogramming methods as data preparation modules in *D2K*

5. System deployment and impact

5.1. Interaction with users

We consulted with the Allstate underwriting division biweekly through the third through fifth months of the nine-month project and weekly through the sixth through ninth month. These consultations and preparation of the visualizations constitute 15% of development time and 25% of overall effort. They were critical in two respects:

1. Assisting the users to understand the exploratory experiments, the classification models, the use of these models in the performance element (interactive decision support tools based on *MineSet*, *Clementine*, and *D2K*), and the data mining process
2. Elicitation of subject matter expertise in Phase I (the first 3 months) on attribute synthesis, data clustering, and semantics of the data model (especially the data dictionary for *ALLVAR* and ambiguities in the classification rule base)
3. Elicitation of user feedback in Phases II and III (the last 4 months) on attribute relevance, selection criteria (comprehensibility of the decision trees, appropriate overfitting tradeoffs, and utility of decision trees with various test set accuracies for different quantization, or bin, granularities)

Most important, it was critical to understand the business objective of the underwriting team: development of an experimental pricing strategy based on several data-driven performance elements (classification models for populations of policies in different lines of business; churn prediction models; interactive query and visualization; and human-readable rules). We found that a flexible, diagrammatic work flow model leading up to visualization and interactive interpretation, as is used in *MineSet* (Kohavi, 1998), *NeuroSolutions* (Principé and Lefebvre, 1998), and *D2K* (Auvil et al., 1999), was most useful to the underwriting team. This allowed us to develop a *sustainable* data mining system that our research group could transfer to theirs.

5.2. *Software reuse*

Table 11 lists, in roughly chronological order, the tasks completed during the One Company project over a period of 10 months; the time investment in each phase in terms of development weeks and full-time employee equivalents (FTE); overall effort; and computational resources. Following Brooks's cautions (Brooks, 1995), we added personnel to the project only when a task could be clearly delineated as new module development (e.g., Java data preparation modules, *MineSet* batch scripts, or the Jenesis component of *D*2*K*). It is also important to distinguish the e*ffort investment* in each task as opposed to the *duration* of each task (development time).

We note especially the amortization of effort due to reuse of *D2K* modules. Data preparation modules achieve the highest rate of reuse (up to 6:1 across projects managed by the NCSA Automated Learning Group), but even machine learning modules average 2:1. This reuse actually reduced the amount of time spent on preprocessing and the efficiency of interpreting results (and developing the infrastructure of the performance element), so that 30% of the time and 40% of the effort could be devoted to inductive learning research and development. In our experience, this rate is atypically high.

5.3. *Deployed decision support applications*

The One Company itinerary was delivered to the Allstate underwriting division over the 10-month development phase (August, 1998–May, 1999) of a project that ran for about 1.5 years. Early consultations on development of *ALLVAR-1* occurred during summer, 1998; final delivery of the performance element and its deployment occurred in mid-September, 1999. The system is currently being used by a new data mining research group in the Allstate underwriting division; furthermore, this "model-driven pricing" group is implementing the pipeline and abstract itinerary described in this paper for related decision support applications, using commodity off-the-shelf KDD packages.

Other NCSA Industrial Partners besides Allstate are currently utilizing the *D2K* rapid application development environment for data mining. Currently, Caterpillar has multiple projects implemented in *D2K*. For one of the projects, Caterpillar is training 300 engineers to use the system by the end of the 1999 calendar year. The Sears Home Service Division is in the process of building a data mining application in *D2K*. When completed, they plan to use this system to deliver decision support to their service sites. D2K is also currently being used to build prototype applications by the Illinois State Government Agencies and by academic organizations, such as Kansas State University, for education and research.

Acknowledgments

Support for this research was provided in part by Allstate Insurance Company and by the office of Naval Research under Grant N00014-00-1-0769. We thank Tilt Thompkins and William M. Pottenger of NCSA for administering the Allstate One Company project and for developing the rule simulations that made the machine learning research reported in this paper possible. In addition, we thank the following NCSA staff members for their contributions to the project: Yuching Ni, NCSA's consultant to Allstate; and our fellow researchers at the NCSA Automated Learning Group (ALG): Loretta S. Auvil, Colleen Bushell, Lisa Gatzke, and David Tcheng. We thank the following ALG students for their assistance with *D2K* and *Jenesis* development: Michael Bach, Russ Bader, Mike Perry, Kristopher Wuollett, Ting-Hao Yang, and Dav Zimak. We thank David E. Goldberg for insightful remarks regarding genetic algorithms for constructive induction and Larry A. Rendell for early discussions on inductive bias optimization. We also thank the director of the Allstate Underwriting Division, Joe Porter, for sharing his expertise and for his weekly participation in experimental discussions and visualization sessions. Finally, we thank the anonymous reviewers for beneficial comments regarding wrappers for performance tuning in KDD, hyperparameter (especially inductive bias parameter) optimization, and distributed computing performance and scalability issues.

Notes

1. In our simplified IBL algorithm, we computed Voronoi cells using the following steps:
    1. Compute relative maxima of cluster density on the map (by histogramming the output of *SOM-PAK*, which gives the winning map coordinate for each training example), thresholded by a *user-specified* value (the minimum number of representative examples—in our case, insurance policies). This technique was also successful for a project on text document categorization described in Section 5.2.
    2. Set the cluster representatives to be these "thresholded peaks".
    3. Compute a Voronoi diagram in one offline pass, using the cluster representatives. By contrast, LVQ (Kohonen, 1990) computes this diagram, and the centers of the classifier decision regions, dynamically.

    Our simplification allows users to determine the desired number of clusters indirectly, by specifying a runtime parameter that, as we have found, is often intuitive to the domain specialist: the *number of representative data points needed to constitute a cluster*.
2. The simulations were conducted on *ALLVAR-2*, a second preprocessed data set, only because certain ambiguities in the rule set resulted from the incomplete data model definition in *ALLVAR-1*. There is every indication that the trends seen in both the cluster and tier-based classifiers hold over both data sets.

References

Aha, D., Kibler, D., and Albert, M. 1991. Instance-based learning algorithms. Machine Learning, 6:37–66.

Auvil, L., Redman, T., Tcheng, D., and Welge, M. 1999. Data to Knowledge (D2K): A Rapid Application Development Environment for Knowledge Discovery. NCSA Technical Report, URL: http://archive.ncsa.uiuc.edu/STI/ALG/d2k.

Benjamin, D.P. (Ed.) 1990. Change of Representation and Inductive Bias. Boston: Kluwer Academic Publishers.

Brooks, F.P., Jr. 1995. The Mythical Man-Month: Essays on Software Engineering, 20th Anniversary Edition. Reading, MA: Addison-Wesley.

Cheeseman, P., Kelly, J., Self, M., Stutz, J., Taylor, W., and Freeman, D. 1988. AUTOCLASS: A bayesian classification system. In Proceedings of the Fifth International Conference on Machine Learning (ICML-88), pp. 54–64.

Cherkauer, K.J. and Shavlik, J.W. 1996. Growing simpler decision trees to facilitiate knowledge discovery. In Proceedings of the Second International Conference of Knowledge Discovery and Data Mining (KDD-96): Portland, OR.

Dempster, A., Laird, N., and Rubin, D. 1977. Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society, 39(Series B):1–38.

Donoho, S.K. 1996. Knowledge-Guided Constructive Induction. PhD Thesis, University of Illinois at Urbana-Champaign (Technical Report UIUC-DCS-R1970).

Dejong, K.A., Spears, W.M., and Gordon, D.F. 1993. Using genetic algorithms for concept learning. Machine Learning, 13:161–188.

Fayyad, U., Piatetsky-Shapiro, G., and Smyth, P. 1996. Knowledge Discovery and Data Mining: Towards a Unifying Framework. In Fayyad, U., Piatetsky-Shapiro, G., and Smyth, P., editors, Advances in Knowledge Discovery Data Mining, pp. 82–88. Cambridge, MA: MIT Press.

Gersho, A. and Gray, R.M. 1992. Vector Quantization and Signal Compression. Norwell, MA: Kluwer Academic Publishers.

Goldberg, D.E. 1989. Genetic Algorithms in Search, Optimization, and Machine Learning. Reading, MA: Addison-Wesley.

Grefenstette, J.J. 1990. Genesis Genetic Algorithm Package.

Haykin, S. 1999. Neural Networks: A Comprehensive Foundation, 2nd edn. Englewood Cliffs, NJ: Prentice Hall.

Hsu, W.H., Welge, M., Wu, J., and Yang, T. 1999. Genetic algorithms for selection and partitioning of attributes in large-scale data mining problems. In Proceedings of the Joint AAAI-GECCO Workshop on Data Mining and Evolutionary Algorithms, Orlando, FL.

Hsu, W.H. 1998. Time series learning with probabilistic network composites. PhD Thesis, University of Illinois at Urbana-Champaign (Technical Report UIUC-DCS-R2063).

Hsu, W.H., Ray, S.R., and Wilkins, D.C. 2000. A Multistrategy Approach to Classifier Learning from Time Series. Machine Learning, 38(1–2):213–236. Norwell, MA: Kluwer Academic Publishers.

Hsu, W. and Welge, M. to appear. Activities of the Prognostics Working Group. NCSA Technical Report.

John, G., Kohavi, R., and Pfleger, K. 1994. Irrelevant features and the subset selection problem. In Proceedings of the 11th International Conference on Machine Learning, New Brunswick, NJ. Morgan-Kaufmann, Los Altos, CA, pp. 121–129.

Jonske, J. 1999. Personal communication. Unpublished.

Kohavi, R., Becker, B., and Sommerfield, D. 1997. Improving simple bayes. Presented at the European Conference on Machine Learning (ECML-97).

Kohonen, T., Hynninen, J., Kangas, J., and Laaksonen, J. 1996. SOM-PAK: The Self-Organizing Map Program Package. Technical Report A31, Helsinki University of Technology, Laboratory of Computer and Information Science, FIN-02150 Espoo, Finland.

Kohavi, R. and John, G.H. 1997. Wrappers for feature subset selection. Artificial Intelligence, Special Issue on Relevance, 97(1/2):273–324.

Kohonen, T. 1990. The self-organizing map. Proceedings of the IEEE, 78:1464–1480.

Koza, J. 1992. Genetic Programming: On the Programming of Computers by Natural Selection. Cambridge, MA: MIT Press.

Kononenko, I. 1994. Estimating attributes: Analysis and extensions of relief. In Proceedings of the European Conference on Machine Learning, F. Bergadano and L. De Raedt (Eds.).

Kohavi, R. 1995. Wrappers for Performance Enhancement and Oblivious Decision Graphs. PhD Thesis, Department of Computer Science, Stanford University.

Kohavi, R. 1998. MineSet v2.6, Silicon Graphics Incorporated, CA.

Kira, K. and Rendell, L.A. 1992. The feature selection problem: Traditional methods and a new algorithm. In Proceedings of the National Conference on Artificial Intelligence (AAAI-92), San Jose, CA. Cambridge, MA: MIT Press, pp. 129–134.

Krishnamurthy, B. (Ed.) 1995. Practical Reusable UNIX Software. New York: John Wiley and Sons.

Kohavi, R. and Sommerfield, D. 1996. MLC++: Machine Learning Library in C++, Utilities v2.0. URL: http://www.sgi.com/Technology/mlc.

Mitchell, T.M. 1997. Machine Learning. New York, NY: McGraw-Hill.

Neal, R.M. 1996. Bayesian Learning for Neural Networks. New York, NY: Springer-Verlag.

Principé, J. and Lefebvre, C. 1998. NeuroSolutions v3.02, NeuroDimension, Gainesville, FL. URL: http://www.nd.com.

Porter, J. 1998. Personal communication. Unpublished.

Quinlan, J.R. 1985. Induction of decision trees. Machine Learning, 1:81–106.

Quinlan, J.R. 1990. Learning logical definitions from relations. Machine Learning, 5(3):239–266.

Russell, S. and Norvig, P. 1995. Artificial Intelligence: A Modern Approach. Englewood Cliffs, NJ: Prentice Hall.

Raymer, M., Punch, W., Goodman, E., Sanschagrin, P., and Kuhn, L. 1997. Simultaneous feature extraction and selection using a masking genetic algorithm. In Proceedings of the 7th International Conference on Genetic Algorithms, San Francisco, CA, pp. 561–567.

Sarle, W.S. (Ed.). Neural Network FAQ, periodic posting to the Usenet newsgroup comp.ai.neural-nets, URL: ftp://ftp.sas.com/pub/neural/FAQ.html

Sterling, T.L., Salmon, J., Becker, D.J., and Savarese, D.F. 1999. How to Build a Beowulf: A Guide to the Implementation and Application of PC Clusters. Cambridge, MA: MIT Press.