1-1-2006

A comparative study of public domain supervised classifier performance on the UCI database

Peter W. Eklund
University of Wollongong, peklund@uow.edu.au

A Hoang
University of Wollongong

A comparative study of public domain supervised classifier performance on the UCI database

Abstract

This paper surveys three classes of public domain supervised learning algorihms and performs comparative analysis of their performance against 29 of the University California Irvine machine learning datasets.

Keywords

comparative, study, public, domain, supervised, classifier, performance, UCI, database

Disciplines

Business | Social and Behavioral Sciences

Publication Details

Eklund, P. W. & Hoang, A. (2006). A comparative study of public domain supervised classifier performance on the UCI database. Australian Journal of Intelligent Information Processing Systems, 9 (1), 1-39.

A Comparative Study of Public Domain Supervised Classifier Performance on the UCI Database

Peter W. Eklund and A. Hoang

School of Economics and Information Systems
The University of Wollongong
Northfields Ave, Wollongong NSW 2522
Australia

Abstract

This paper surveys three classes of public domain supervised learning algorithms and performs comparative analysis of their performance against 29 of the University California Irvine machine learning datasets. These classifiers represent three dimensions of classifier: decision trees, neural networks and rule-based classifiers. The study uses data visualisation techniques to justify the performance characteristics of the surveyed algorithms and makes some general recommendations about the selection of public domain classifiers relative to the data properties we examine.

1 Introduction

Knowledge and Data Discovery (KDD) draws on many fields including databases, statistics, visualization, and artificial intelligence (AI). In the last fifteen years, the number of KDD tools and techniques have grown steadily. KDD methods embrace statistical approaches like linear programming, regression analysis and principal component analysis as well as AI-based techniques such as decision trees, rule discovery algorithms, neural networks, and genetic algorithms. The critical AI technologies have been broadly identified as machine learning and knowledge-based systems [10], however in the machine learning literature, supervised classification algorithms[1] have a decisive impact on KDD performance.

A supervised machine learning algorithm is a computer program that is presented with a number of attribute features demonstrating a particular outcome class as an example. The program is tasked with generating a decision procedure called a "classifier". The classifier is used to predict the outcome class of an unseen example on the basis of observing training examples. The measure of the classifiers performance is how well it can successfully predict class outcomes on unseen examples. A key question during the last decade concerns supervised classification algorithms embedded in KDD tools. Which of these algorithms constructs classifiers that most consistently predict class outcomes with high degrees of accuracy?

Naturally, a technique may be suited to one problem domain and not another and there is therefore no universal algorithm. This is the reason why choosing a classifier algorithm is still something of an art [11]. This also explains why researchers focused on developing new supervised classifier techniques or improving existing classification tools rather than comparative studies.

Much has been done to provide benchmarking infrastructure for learning algorithms but more work is required. The establishment of a public repository of Machine Learning Databases in University of Irvine, California [17], the Espirit Project 5170 Stat-Log (1991-1994) [14, 15, 3], and events such as the Information Exploration Shootout [13] provide the basis for this benchmarking infrastructure. However, the diversity of database characteristics as well as the increasing number of newly created classifier algorithms, particularly those outside the Espirit Stat-Log Project [15], demands continuous comparative studies across domains and algorithms[2].

A comparative study cannot be complete, for instance the results of StatLog tell us little about the relative performance of algorithms outside its scope. Likewise this study is incomplete. Additional performance studies often accompanying software, however, these typically make comparisons between two or three algorithms [19, 20, 7, 26, 24, 22, 23] and are often selected to recommend the package they accompany.

The lack of information on comparative studies of public domain supervised classification algorithms is the main motivation of our research. It also provides the basis for the evaluation of public domain resources in terms of their relevance to the benchmark tasks they support. More importantly, we combine our comparative analysis with a

[1] As opposed to un-supervised or concept clustering algorithms

[2] The paucity of comparative analysis extends to public domain tools freely accessible via the Internet. Only 3 public domain supervised algorithms are included in the StatLog project: CN2, LVQ, and the Radial Basis Function Network (RBFN) [3].

data analysis and visualization study. This provides a guide to select and match the appropriate classifier to the problem domain. The objective of this study is therefore focused on the question: how well do public domain supervised classifiers perform on different data? The second question is: given data with certain characteristics and features, which supervised learning algorithm is most appropriate?

StatLog provides the most comprehensive comparative study of different supervised machine learning algorithms, i.e. symbolic, neural and statistical classification algorithms. The aim of StatLog is to obtain a set of rules characterizing the applicability of one of twenty algorithms on a dataset with given characteristics/features [3].

The measures used in Statlog to characterise data are: *simple*, *statistical*, and *information based*.

- *simple measures* are the number of examples, classes, attributes, the proportion of binary attribute values, the number of unknown attribute values;
- *statistical measures* include standard deviation ratios of the algorithms performance over several runs and the mean values of correlations between attributes;
- *information based measures* are entropy criterion such as the mean mutual information gain by selecting class and attributes.

Our experiments have a similar purpose to StatLog and should therefore consider similar measures.

In addition, another dimension of the data is the source from which it was derived. Data is obtained from various sources: medical, biological, chemical, image processing etc; and according to its origins, functional dependencies between attributes are more or less likely to be present. The remaining sections of this paper are organized as follows:

1. Section 2 describes the public domain supervised machine learning algorithms used in this study;
2. Section 3 presents information on the data used in the study;
3. Section 4 consists of three parts. The first describes the experimental method used. The second and third present the results of measurements obtained from the experimental trials and a statistical analysis of the results;
4. Section 5 presents the results of data analysis using visualization and feature selection techniques;
5. Section 6 discusses the experimental results. In particular, it provides explanations for performance differences between the various classifier algorithms;
6. Section 7 gives a brief conclusion about what has been achieved and (for the reviewer) what remains to be done.

2 Supervised Learning Algorithms

The duration of our study necessitated several restrictions on the algorithms tested. Firstly, because of the academic nature of the research, algorithms are restricted to those embedded within software in the public domain. More acutely, all algorithms should be accessible via the Internet or easily accessible in the public domain. Secondly, algorithms should be compatible in the way they report classification accuracy. Finally, the algorithms tested should be implemented in C or C and capable of running on standard Unix platforms without the aid of special hardware[3].

LMDT, SNNS, C4.5, C4.5 rules, CN2, ITI, Nevprop, LVQ, OC1 and three TOOLDIAG classifiers where selected. These software packages represent four types of supervised machine learning classifiers: Decision Trees, Neural Networks, Rule-based and Instance-based approaches.

2.1 C4.5

C4.5 [25] is the most popular decision tree classifier. C4.5 induces decision trees[4] in a top-down manner, using the divide-and-conquer method. The C4.5 learning algorithm is described in Algorithm 1.

C4.5 uses gain ratio criterion to score tests. There are three types of test used by the program. The first is where the decision tree is split on each attribute value, with one outcome and branch for each possible value of that attribute. The second is based on analysis of discrete attribute values. Possible attribute values are allocated to a variable number of groups with one outcome for each group rather than one outcome for each value. The last applies to continuous numeric attribute values. This is a binary test (with outcome $A \leq Z$ and $A > Z$) based on comparing the value of A against some threshold value Z. To cope with the problem of missing values, C4.5 uses two additional values: the probability the attribute is known or unknown. An error-based tree pruning technique [28] is

[3] This steers the study away from parallel learning algorithms and environments.

[4] A decision tree consists of internal (or decision) nodes and leaf nodes. An internal node of a tree specifies a univariate test of an attribute (or feature value), with each outgoing branch corresponding to the result of this test. A leaf node represents the classification to be assigned to an example. To classify a new example, a path from the root of the decision tree to a leaf node is traced. At each internal node, the branch corresponding to the value of the attribute tested is followed. The class at the leaf node represents the class predicted for that example

used to avoid overfitting training data[5]. From the resulting decision tree, C4.5 also builds a rule classifier that is approximately as accurate as the pruned tree. The steps in this process are as follows:

1. each path from the root of an unpruned tree to a leaf gives one initial rule. The left-hand side of the rule contains the conditions established by the path and the right-hand side the class at the leaf node;
2. each such rule is simplified by removing conditions that are unhelpful in discriminating the nominated class from other classes. This is done using a pessimistic estimate of the accuracy of the rule;
3. for each class in turn, all the simplified rules for the class are filtered to remove rules not contributing to the accuracy of the set of rules as a whole;
4. the sets of rules for each class are ordered to minimize false positive errors and a default class is chosen.

2.2 ITI – Incremental Decision Tree Induction

ITI [22, 8] (Incremental decision Tree Induction) is available by ftp from `ftp.cs.umass.edu` in the directory `/pub/iti`. ITI does not require any special data format, working well on data with (or without) missing values.

The ITI incremental decision tree induction algorithm is built around the tree revision mechanism [22]. Tree revision assumes that every possible test at a decision node has exactly two possible outcomes: this means that the decision tree produced is always binary. At every decision node, the frequency counts for each outcome-class combination for a symbolic variable, or the list of values observed in the instances at that node in sorted order, are kept and updated as necessary in order to change the test used at a decision node. The ITI algorithm is described in Algorithm 2. ITI incrementally incorporates a new instance into the existing decision tree by passing it down proper branches, typically until a leaf is reached. Because passing an instance through a node causes information kept at that node to be changed and consequently invalidates the basis on which the installed test at that node was selected, ITI marks as stale any decision node through which the new instance has passed. It makes this tag so that a later visit to the node can re-install a new test. ITI allows the use of the same gain-ratio metric as used in C4.5 or other direct metrics such as expected number of tests, the number of leaves, the minimum description length, the expected classification expense, the expected misclassification cost [22] or the Kolmogorov-Smirnoff distance [23].

ITI training can run in several modes. In the default mode each instance presented is incorporated into the decision tree, which is then immediately restructured as necessary so every decision node has its most desired test installed. Another mode is the error correction mode for a fixed pool of training instances. ITI cycles through the candidate pool repeatedly, removing any incorrectly classified instances and incorporating them into the decision tree. It does this until the tree does not misclassify any instance remaining in the pool. When a batch of instances is received one at a time, *lazy mode* is the appropriate training method. This adds each instance to the tree without later revisiting the tree. After all the instances from the batch have been incorporated, a single call to the procedure for ensuring the best test is installed at each node will bring the tree into its final form. The three training modes ensure that ITI can run on a sliding scale from purely incremental to non-incremental learning modes.

To avoid overfitting, ITI uses a virtual pruning technique based on the minimum description length principle [28]. The default approach considers whether each subtree can be represented more compactly by a leaf with a default class and a list of exceptions: each exception is an index into the list of instances and an indication of its non-default class label. Pruning any subtree is performed only in the form of marking the root decision node as pruned and nothing is discarded. This technique is useful for an incremental induction approach because we can add or remove the mark at any time.

The tree revision mechanism also allows ITI to revise an existing tree according to a direct metric in the form DMTI (Direct Metric Tree Inducer) [22]. This uses various metrics (mentioned above) to revise the existing tree inexpensively. DMTI often produces dramatic improvements over ITI.

2.3 OC1 – Oblique Classifier

OC1 (Oblique Classifier) is a system for the induction of oblique decision trees from examples [26]. The software can be obtained from `ftp.cs.jhu.edu` in directory `/pub/oc1`. Oblique decision trees are trees in which each node may contain a multivariate test on attributes of the data. OC1 only works in domains where all attributes are numeric-valued and the class labels integer-valued. Attributes with missing values are also allowed.

OC1 follows the top-down approach of decision tree induction (also called the greedy divide and conquer method) similar to C4.5. The main differences between the C4.5 algorithm (see Algorithm 1) and OC1 algorithm are:

[5] Pruning a decision node consists of removing the subtree rooted at that node, making it a leaf node, and assigning it the most common classification of the training examples affiliated with that node. Nodes are removed only if the resulting pruned tree performs no worse than the original over the holdout subset (20% of training cases are selected at random for this holdout set). Pruning continues until further pruning is harmful (i.e., decreases accuracy of the tree over the holdout set).

Inputs:
Let T be a set of training examples.
Let *Attributes* be the set of attributes that may be tested by the learned decision tree.
Outputs:
A decision tree D that correctly classifies the given example.

procedure C4.5(T, *Attributes*)
Create a Root node for the tree.
if all examples in T belong to one class **then**
Set the *Root* to be a single leaf node with label = class label of all examples in T.
else
if Attributes is empty **then**
Set the *Root* to be a single leaf node with label = most common value of attribute, whose value is to be predicted by the tree.
else
Consider all tests that divide T into two or more subsets.
Score each test according to how well it splits the examples.
Assign A the attribute from *Attributes* that scores the highest.
for each possible value v_i of A **loop**
Add a new subtree below *Root*, corresponding to the test $A = v_i$.
Let Tv_i be the subset of T that have value v_i for A.
C4.5(Tv_i, *Attributes* – {A}).
end loop
end if
end if
return *Root*.
end

Figure 1: The C4.5 training algorithm.

```
Inputs:
        Let T be a set of training instances.
Outputs:
        A decision tree D that correctly classify the given example.

procedure ITI(T)
    Let D be an empty tree.
    for all instances x in T loop
        if D is empty then
            Set D to be a leaf node with class label = class label of x.
        else
            Based on the attribute-values in x, pass it down the tree until
            a leaf is reached.
            if label of x = class label of the leaf then
                Add x to the set of instances saved at the node.
            else
                Set the leaf to be a decision node. The attribute to be tested
                at this node is "best" according to the attribute-selection metric.
                Send all instances saved at this node one down
                according to the new test installed at the node.
                Visit each stale node recursively to select and install the best test
                at these nodes.
            end if
        end if
    end loop
end
```

Figure 2: Incremental Tree Induction algorithm (ITI).

in C4.5, a test at a node checks the value of a single attribute and splits the attribute into all of its values when the attribute is symbolic or discrete integer. When the attribute is numeric, the tests have the form $A > Z$ where A is one of the attributes of an example and Z is a constant (also called a threshold value). This type of test is called *univariate* and this class of decision trees is called *axis-parallel* because the test of each node is equivalent to an axis-parallel splitting hyperplane in the attribute space.

The algorithm considers all possible axis-parallel hyper-planes and chooses the one that "best" splits the training examples. OC1 however uses another type of test at its decision node. This test checks a linear combination of attributes and has the form:

$$\sum_{i=1}^{d} a_i \cdot x_i + a_{d+1} > 0$$

where x_i are real valued attributes a_i are real-valued coefficients. This type of test is called *multivariate* and this class of decision trees is called *oblique* because the test of each node is equivalent to an oblique splitting hyperplane in the attribute space. Because it is impossible to find all possible oblique splitting hyperplanes, OC1 uses a special way of finding "the best" split. The basic algorithm for finding the best split at each node of a decision tree is summarized in Algorithm 3.

OC1 uses a special perturbation algorithm to perturb the hyperplane H which halts when the split reaches the minimum of the impurity measure. OC1 works on a wide range of impurity measures such as Information Gain, The Gini Index, The Twoing Rule, Max Minority, Sum Minority and Sum of Variances [26]. OC1 uses randomization techniques to escape local minimum, when no perturbation of any single coefficient hyperplane will decrease the impurity measure. This technique perturbs the hyperplane with a random vector and re-starts the perturbation algorithm with a different random initial hyperplane. If after n random jumps (the value of n is set to 5 by default and can be specified by the user) fails to improve the impurity, OC1 halts and uses H as the split for the current tree node. To avoid over-fitting training data, OC1 uses Breiman et al's Cost Complexity [16] (also called Error Complexity or Weakest Link) pruning as the default method. The cross validation process in OC1 is costly with respect to computing time and OC1 is unable to run on very large datasets[6] . OC1 is supplied with all necessary facilities for knowledge reuse and classification from unlabeled samples.

2.4 LMDT -Linear Machine Decision Tree

LMDT (Linear Machine Decision Tree) [7, 24] induces multi-class decision trees with multivariate tests at internal decision nodes. LMDT can be obtained through contact

[6] In this study it would not run on the UCI *Letter Recognition* database.

Inputs:
Let T be a set of training examples.
Outputs:
The best oblique parallel split of T.

procedure BEST_SPLIT(T)
Find the best axis-parallel split of T.
Let I be the impurity of this split.
Let R be the number of jumps defined by the user.
repeat R times
Choose a random hyperplane H.
(For the first iteration, initialize H to be the best axis-parallel split)
Step 1: **until** the impurity measure does not improve **do**
Perturb each of the coefficients of H in sequence.
Step 2: **repeat** at most J times
Choose a random direction and attempt to perturb H in that direction.
if this reduces the impurity of H **then**
go to Step 1.
end if
Let I_1 be the impurity of H. If $I_1 < I$ then set $I = I_1$.
return the split corresponding to I.
end

Figure 3: The basic algorithm for finding the best oblique split of OC1.

with its authors `brodley@ecn.purdue.edu`. LMDT does not require any special data format and can work on any type of data with (or without) missing values.
LMDT builds a decision tree in the well-known top-down manner [24]. The top-level algorithm is described in Algorithm 4. The main differences between LMDT and other decision tree induction methods are: instead of selecting a univariate test for a decision node based on a heuristic measure, such as information gain (used by C4.5), LMDT trains a linear machine which then serves as a multivariate test for the decision node. A linear machine is a multi-class linear discriminant function which itself classifies the instance. The class name is the result of the linear machine test, hence there will be one branch for each possible class at the node. Linear machine decision trees can be understood as a hybrid representation, produced by a combination of two methods, one for learning decision trees and the other for learning linear discriminant. functions.

A linear machine is a set of discriminant functions that are used collectively to assign an instance to one of the R classes. Let Y be an instance description, also known as a pattern vector, consisting of a constant threshold value of 1 and the numerically encoded features through which the instance described. Each discriminant function $g_i(Y)$ has the form $W_i^T Y$, where W is a vector of adjustable coefficients: also known as weights. A linear machine infers instance Y to belong to class i iff $(\forall i, i \neq j)$, $g_i(Y) < g_j(Y)$. If there is no unique maximum, the classification is undefined.

To train a linear machine, one adjusts the weight vectors W of the discriminant function g in response to any instance that the linear machine might misclassify. This is done by increasing the weights of W_i, where i is the class to which the instance belongs, and decreasing the weights of W_j, where j is the class to which the linear machine erroneously classifies the instance. If the instances are separable by a linear machine, then the above training procedure will find a solution linear machine in a finite number of steps [24].

LMDT uses a special method called "thermal perceptron method" for adjusting weights while training the linear machine; this ensures stable behaviour even when instances are not linearly separable. During the training of a linear machine, LMDT simultaneously eliminates variables not contributing to classification at a node. Because the linear machine algorithm requires all variables be ordinal (integer or real-valued), LMDT encodes symbolic variables as ordinal. the encoded symbolic and ordinal variables are normalized automatically at each node.

Scaling is accomplished for each encoded variable by mapping it to a standard normal form, i.e., zero mean and unit standard deviation. Missing values are mapped to 0, corresponding to the sample mean of the corresponding encoded variable. All encoding information is computed dynamically at each node and retained in the tree for the purpose of classifying instances. LMDT is computationally costly. The program has no hold-out validation mechanism

```
Inputs:
    Let T be a set of training examples.
Outputs:
    A decision tree D that correctly classifies the given example.

procedure LMDT(T)
    Create a Root node for the tree.
    if all examples in T belong to one class then
        Set the Root to be a single leaf node with label = class label of all
        examples in T.
    else
        Set the Root to be a decision node.
        Train a linear machine to construct a test to be installed at this decision node.
        if the test partitions instances in T in two or more subsets then
            for each subset T_i loop
                Add a new tree branch below Root, corresponding to the outcomes
                of the test.
                LMDT(T_i).
            end loop
        else
            Set the Root to be a leaf node with label = class label of the most frequently
            occurring class in T.
        end if
    end if
    return Root
end
```

Figure 4: The LMDT training algorithm.

and so requires external splitting of data for validation purposes.

2.5 LVQ-Learning Vector Quantization

LVQ (Learning Vector Quantization) [12] is available by ftp at `cochlea.hut.fi` in directory `/pub/lvq_pak` . The package contains all necessary programs[7] for the correct application of LVQ algorithm in arbitrary statist ical classification or pattern recognition tasks. LVQ requires real- valued attribute values of data but works well on integer-valued data space with (or without) missing values.

LVQ algorithms try to established a number of codebook vectors into the input space to approximate various domains of the input vector by their quantized values. The labeled samples are fed into a generalization module that tries to find a set of codebook vectors representing knowledge about the class membership of the training set.

In LVQ, different learning procedures can be selected to produce codebook vectors such as LVQ1, LVQ 2.1, LVQ3 and OLVQ. The LVQ1 learning algorithm is described in Algorithm 5. During the learning process, the codebook vector nearest to the input training instance is updated to correctly classify the training instance.

The LVQ2.1 learning procedure differs from LVQ1 in that it simultaneously updates two codebook vectors that are nearest neighbours to an input vector x, one of which must belong to the correct class and the other to an incorrect class. x must fall into a zone of values called a 'window' defined around the midplane of these two codebook vectors using Euclidian distances of x from them. The learning equations are:

$$m_i(t+1) = m_i(t) - \alpha(t)[x(t) - m_i(t)],$$
$$m_j(t+1) = m_j(t) + \alpha(t)[x(t) - m_j(t)]$$

where m_i and m_j are the two closest codebook vectors to x, whereby x and m_j belong to the same class, while x and m_i belong to different classes respectively.

The LVQ3 learning procedure ensures that the m_i continue approximating the class distribution, at least roughly by including some correction. It uses the same equation and conditions above but requires x to fall into the window.

$$m_k(t+1) = m_k(t) + \varepsilon\alpha(t)[x(t) - m_k(t)]$$

for $k \in \{i, j\}$, if x, m_i, m_j belong to the same class and $0.1 < \varepsilon < 0.5$.

The optimized LVQ1 (OLVQ1) procedure uses individual learning rate $\alpha_i(t)$ assigned to each m_i.

[7] Four options for the LVQ algorithms, namely LVQ1, LVQ2.1, OLVQ1 and LVQ3 are available together with many auxiliary programs

Inputs:

Let m_i be the set of untrained codebook vectors which are properly distributed among classes.

Let *rlen* be the number of learning cycles defined by the user.

Outputs:

A trained set of codebook vectors mi which correctly classify the given example.

procedure LVQ1

Let $c = argmin_i\|x - m_i\|$ define the nearest m_i to x denoted by m_c.

Let $0 < \alpha(t) < 1$ and $\alpha(t)$ may be constant or decrease monotonically with time.

Let the $m_i(t)$ represent sequences of the m_i in the discrete-time domain.

while learning cycles < *rlen* **loop**

for each input sample $x(t)$ **loop**

if $i \neq c$ **then**

$m_i(t+1) = m_i(t)$.

else

if x and m_c belong to the same class **then**

$m_c(t+1) = m_c(t) + \alpha_c(t)[x(t) - m_c(t)]$.

end if

if x and m_c belong to the different class **then**

$m_c(t+1) = m_c(t) - \alpha_c(t)[x(t) - m_c(t)]$.

end if

end if

end loop

end loop

end

Figure 5: The LVQ1 learning algorithm.

Let $c = argmin_i\|x - m_i\|$. Then,

$$m_c(t+1) = \begin{cases} m_c(t) + \alpha_c(t)[x(t) - m_c(t)] & \text{if } x \in C \\ m_c(t) - \alpha_c(t)[x(t) - m_c(t)] & \text{if } x \notin C \\ m_i(t+1) = m_i(t) & \text{for } i \neq c \end{cases} \quad (1)$$

$\alpha_i(t)$ is determined by the recursion,

$$\alpha_c(t) = \alpha_c(t-1)/(1 + s(t)\alpha_c(t-1))$$

where $s(t) = +1$ if the classification is correct otherwise -1.

LVQ is as fast as C4.5 but requires considerable effort from the user in order to obtain high classification accuracy. The main reason is that LVQ requires the user to specify many of the learning parameters, and these vary from task to task (the number of code-book vectors, the distribution and balancing distribution of the codebook vectors among classes, tuning the induced classifier). This parameter specification problem is sure to be one of the contributing factors in LVQ's performance in this experiment. On the other hand, LVQ provides all necessary facilities for knowledge reuse, class-ification of unlabeled examples and is a valuable tool for first time users.

2.6 CN2

CN2 [19, 20] is a rule induction system for inducing concept descriptions from examples. The objective of its authors is that the system should be able to classify new examples accurately, even in the presence of noise. The set of produced rules should be simple and the rule generation process efficient in the size of the datasets. The representation for rules output by CN2 is an antecedent-consequent rule[8] also known as a "decision list". CN2 works well with symbolic, integer-valued, real-valued or mixed continuous-discrete data values. It also works with or without missing values.

The CN2 algorithm [20] consists of two main procedures: a search algorithm performing a beam search for a good rule and a control algorithm for repeatedly executing search. The control algorithm is summarized in Algorithm 6 and the beam search algorithm in Algorithm 7. The control algorithm works in an iterative fashion, each iteration searching for a covering for a large number of examples of a single class C and few other classes. The

[8] To use the induced rules to classify new examples, each rule is tried until one is found whose conditions are by the example being classified. The resulting class prediction of this rule is then assigned as the class of that example. Thus, the ordering of rules is important. If no induced rules are satisfied, the final default rule assigns the most common class to the new example.

```
Inputs:
        Let E be a set of training examples.
        Let RULE_LIST be the empty list.
    Outputs:
        The set of rules RULE_LIST.

procedure CN2(E)
    repeat
        Let BEST_CPX be Find_Best_Complex(E).
        if BEST_CPX is not nil then
            Let E' be the examples covered by BEST_CPX.
            Remove from E the examples E' covered by BEST_ CPX.
            Let C be the most common class of examples in E'.
            Add the rule "if BEST_CPX then class = C" to the end of RULE_LIST.

        end if
    until BEST_CPX is nil or E is empty.
    return RULE_LIST.
end
```

Figure 6: The CN2 training algorithm.

complex[9] must be both predictive and reliable, as determined by CN2s evaluation functions. Having found a good *complex*, those examples it covers[10] are removed from the training set and the rule "if <complex> then predict *C*" is added to the end of the rule list. This process iterates until no more satisfactory complexes can be found. Two heuristic tests are used during the learning process to determine whether the complexes, found during search, are both "good" and "reliable". An information-theoretic entropy measure is used to evaluate the complex quality and the Laplace expected error estimate to test whether a complex is significant.

CN2 deals with continuous attributes by dividing the range of values of each attribute into discrete sub-ranges. Tests on such attributes examine whether a value is greater (or less than equal) the values at sub-range boundaries. Unknown attribute values are simply replaced with the most commonly occurring value for that attribute in the data.

CN2 is not as fast as C4.5 or LVQ but faster than ITI, LMDT, and OC1. The program can run in interactive or batch mode with all default learning parameters. Help facilities are also available. One small detail should be mentioned here, CN2 has a special lexical requirement of data and attribute file formats and consequently inexperienced users might have some difficulty in domains with symbolic feature values. Another problem is CN2, at least in the public domain version, has no facility either for the prediction task on unlabeled examples or for reusing the rule set.

2.7 TOOLDIAG

TOOLDIAG [9] can be obtained from `ftp.uninova.pt` in the directory `/users/tr/soft`. It is a collection of methods for statistical pattern recognition. This toolkit's main application is classification and it is limited to multi-dimensional continuous features without missing values. No symbolic features are allowed. TOOLDIAG is implemented in the C programming language and consists of a set of tools, providing the user with many facilities for data analysis. The main facilities are:

Feature selection: This is a strong component of TOOLDIAG. This helps the user reduce the dimension-ality of the vector, of the samples (that characterise a process state) to a reasonable size by selecting those features relevant to the classification task;

Feature extraction: Available features may be combined to create new features with lower dimensionality;

Error estimation: With the reduced feature set, the perfor-mance of a classifier based on these features is forecast;

Induction of different classifiers: From the analyzed dataset, five types of classifier can be produced: *k*-Nearest Neighbor; Quadratic Gausian; Radial Basic Function Network; Parzen window and Q* classifier. The three classifiers used in this experiment: k-Nearest Neighbor, Radial Basis Function Network, and the Q* algorithm are described below;

[9] The basic test on an attribute is called a *selector*. A conjunct of selectors is called a complex. A disjunct of to the new example. complexes is called a *cover*.

[10] We say that an expression (a selector, complex, or cover) *covers* an example if the expression is true of the example

```
procedure Find_Best_Complex(E)
    Let the set STAR contain only the empty complex.
    Let BEST_CPX be nil.
    Let SELECTOR be the set of all possible selectors.
    while STAR is not empty loop
          Specialize all complexes in STAR as follow:
          Let NEWSTAR be the set {x ∩ y|x ∈ ST AR, y ∈ SELECTOR}.
        Remove all complexes in NEWSTAR that are either in STAR (i.e.
        the unspecialized ones) or are null (i.e. big = y ∩ big = n ).
          for every complex C_i in NEWSTAR loop
              if C_i is statistically significant when tested on E and better than
              BEST_CPX according to user-defined criteria when tested on E then
                    Replace the current value of BEST_CPX by C_i.
                end if
          end loop
          repeat
                Remove worst complexes from NEWSTAR.
          until size of NEWSTAR is ≤ user-defined maximum.
          Let STAR be NEWSTAR.
    end loop
    return BEST_PCX.
end
```

Figure 7: The procedure Find_Best_Complex.

Data visualization: The distribution of classes in the feature space can be visualized in the $x - y$ plane or in a three-dimensional space with the help of a non-linear mapping technique. The behaviour of the estimated error in respect to the number of selected features can also be plotted. Furthermore, the linear correlation between two different features can be tested and visualized;

Interface to other program packages: Analyzed data can be normalized (linear to [0, 1]: zero mean, unit variance) or split into training and testing sets. In addition, data can also be passed to other programs such as LVQ and SNNS[11].

The Nearest Neighbor algorithm is one of the most venerable algorithms in Machine Learning [5]. In its simplest versions, the entire training set is stored in memory. To classify a new example, the Euclidean distance (possibly weighted) is computed between the example and each stored training example. The new example is assigned the class of the nearest neighboring example. Better classification accuracy can often be achieved by using more than the first nearest neighbor to classify a query [5, 28]. The first TOOLDIAG classifier used in the experiment is k-Nearest Neighbor with $k = 5$ (*K5* is used for the sake of brevity).

The Radial Basis Function Network (RBF-Network) [28, 29, 9] is the second TOOLDIAG classifier used in this experiment (abbreviated *RBF*). RBF is a classifier paradigm based on artificial neural networks. RBF can be regarded as the graphic notation for the generalized solution to the regularization problem. RBF consists of three layers: a d-dimensional input layer, where d is the dimension of the feature space, a hidden layer, and an output layer with one output unit for each class. The hidden layer is formed by a set of reference vectors representing the underlying distribution function. In the generalized form, the number of hidden units is considerably smaller than the number of data points. Under these conditions, the regularization network can be interpreted as the solution to the multivariate interpolation problem given by a linear superposition of a set of basis functions represented by a subset of the data points.

For every vector at the input layer, RBF supplies the corresponding response at its output, i.e., for each class separately. By comparing the output activation for all classes a classification decision can be made. This input-output mapping is learned before the classification run. Learning, in this context, is equivalent to finding a surface in a multi-dimensional space providing the best fit to the training data, i.e. the best possible approximation to the true probability density function of the underlying distribution. The parameters to be learned are: linear weights associated with the output layer; position of the hidden unit (reference vectors) and; spread of the hidden unit (basis) function.

Two basic learning strategies are used in RBF. The first is supervised learning of all parameters, e.g., on the basis of the error-correction algorithm (a gradient-descent procedure): a generalized form of the well-known Least-Mean-Square algorithm. The second strategy determines the position and spreads of the basis functions first, and

[11] Details of LVQ and SNNS are described below.

then calculates the linear weights of the output layer by applying the reg- ularization solution; the centers of the hidden units may be found by using a clustering algorithm. In our experiment, the set of reference vectors is clustered, the number of hidden units out of the entire set of training vectors is 20% and training algorithm regularization with parameters in the range 0.5 ... 2.0. The variance of each hidden unit is individually adjusted by a factor of 0.1 ... 0.3.

The last of the TOOLDIAG classifiers used is the Q* (Q_STAR) algorithm. This algorithm clusters each class of samples. Each class may have several clusters and each cluster is represented by a prototype. No assumption about the probability distribution of the data is necessary. The algorithm works iteratively. It auto-classifies the data, using the actual set of prototypes. A 1-nearest neighbor is used for classification that attributes the class of the closest prototype to the class sample. If the classification is unsatisfactory, new prototypes are dynamically created. Positively classified samples influence the nearest prototype by dislocating it to the mean (or median) of the samples. If no more creations of prototypes occur (or dislocations of prototypes), the self-organization of prototypes stops. The Q* algorithm has then learned to classify correctly all existing training samples via the dynamically created prototypes. As a result of the learning process, the totality of all samples is compressed to a small set of representative prototypes. This classifier and the LVQ algorithm (described above) behave in a similar way in the sense that the learned data is represented by prototypes.

TOOLDIAG runs in interactive mode and suggests default values for learning parameters. By doing so, it reduces effort for the first time user. For datasets with many attributes or a large number of examples, the feature selection process is rather slow and the program is unable to run feature selection on very large datasets, e.g., the Letter Recognition dataset.

Another problem is knowledge reuse. The main purpose of TOOLDIAG is data analysis and the question of classifier induction is in this sense less important. However, it is useful to have some facilities to audit the classification process or reuse a classifier at a later date. TOOLDIAG does not allow this. As we will see later in our analysis, TOOLDIAG can be used in interesting ways for both data analysis and in developing characteristic performance measures.

2.8 SNNS

SNNS (Stuttgart Neural Network Simulator) [29] is a comprehensive package for the simulation of neural networks and consists of four components: (i) simulator kernel; (ii) graphical user interface; (iii) batch simulator version `snnsbat`; and (iv) network compiler `snns2c`. The graphical user interface XGUI, built on top of the kernel, gives a graphical representation of the neural network and controls the kernel during the simulation run. In addition, the graphical user interface can be used to directly create, manipulate, and visualize neural nets in various ways. Complex networks can be created quickly and easily. SNNS is implemented in ANSI-C programming language, obtained from `ftp.informatik.uni-stuttgart.de` in the directory `/pub/SNNS`. SNNS works on datasets with numeric feature values and can be used for continuous prediction tasks.

SNNS provides 32 different learning algorithms based on the network topology defined by the user. In this experiment, the network used is a fully-connected feed-forward neural network with the most common vanilla-backpropagation learning algorithm. The topology of the net depends on the task performed and the intention of the user. This will be described in a later section but the standard topology consists of three layers: input, hidden, and output layers. Each layer consists of a number of units and the units of every layer are connected to all other units of other layers. Connections (links) are weighted in the sense that each link is assigned a weight initialized by random values between 0 and 1. Training a feed-forward neural network consists of two phases: (i) *forward propagation*, an input pattern is presented to the network. The input is then propagated forward in the net until activation reaches the output layer. (ii) *back propagation*, the output of the output layer is compared with the teaching input. The error δ_j between the output o_j and the teaching input t_j of a target output unit j is used, together with the output o_i of the source unit i, to compute the necessary changes to links w_{ij} . To compute the error for inner units for which no teaching input is available (units of the hidden layer(s)), the errors of the following layer, which are already computed, are used in the formula:

$$\Delta w_{ij} = \eta \delta_j o_i$$
$$\delta_j = \begin{cases} f_j'(net_j)(t_j - o_j) \text{ , if } j \text{ is an output unit} \\ f_j'(net_j)\sum_k \delta_k w_{jk} \text{ , if } j \text{ is a hidden unit} \end{cases} \quad (2)$$

where :

η learning factor (a constant),

δ_j error (difference between the real output and the teaching output) of unit j,

t_j teaching output of unit j,

o_i output of the preceding unit i,

i index of a predecessor to the current unit

j with link w_{ij} from i to j,

j index of the current unit,

k index of a successor to the current unit j with link w_{jk} , from j to k.

SNNS is the most attractive neural network software we know of in the public domain. However, SNNS has no built-in data normalization and validation (hold-out mechanisms) and so it requires considerable time and effort preparing data. Under the conditions of the experiment, the learning process is performed via a graphical user interface visually conditioned by the user.

2.9 Nevprop

Nevprop (Nevada backpropagation) [6] is a feed-forward backpropagation multi-layer perceptron simulator.

Nevprop is controlled by either an interactive character-based interface, or command-line arguments. Nevprop was motivated by an interest in improving the prediction and classification accuracy of mathematical models derived from epidemiological health care databases. The software is freely available from `ftp.scs.unr.edu` in the directory `/pub/cbmr/nevpropdir/`. Nevprop is implemented in the C programming language and works on domains with numeric data with (or without) missing values. Nevprop can be used for continuous prediction tasks.

Like SNNS, Nevprop uses the artificial neural network approach to build knowledge structures. Each layer in the three layer input, hidden and output layers consists of a number of units. Units of layers are linked to other units and connections (links) specified by weights. The learned structure is represented by the topology of the net and the weights of links. Detail about the learning algorithm used in the experiment is similar to that used in SNNS described above. Specific characteristics implemented in Nevprop are that it has many auxiliary built-in mechanisms for automatic training and data preparation. Firstly, Nevprop allows the user to choose one among several ways of imputing missing values (as mean, median or random) and saving this result for later use. If the data input is not of the same scale, causing ineffectiveness of weight randomization, it can be standardized linearly to mean zero and units of standard deviation, or in range -0.5 to +0.5. Nevprop also allows automatic stop training if this process does not improve classification.

3 Data Description

Twenty-nine task databases were selected from the UCI repository of Machine Learning Databases as the basis of our comparison (see `www.uci.edu`). These data are a representative range of domains and data characteristics selected particularly to explore biomedical and natural scientific domains. A brief description of the properties of the data is presented in Table 1. The selection criteria insist that the data be drawn from domains in which there are examples of training data with the following properties;

- only symbolic attributes;
- mixtures of symbolic and ordinal (integer or floating point) attribute values;
- only ordinal attribute values: integer, floating point or both.
- complete (no missing values);
- incomplete (data in which missing values are present).

The data represent pre-specified training data derived from a variety of sources: medical, biological, or sociological experimental domains. Some have several or many missing values while others are complete. These data also differ from each other in the magnitude of examples and number of predictive feature values. Information about the maximal reported classification accuracy from the literature are taken from the UCI repository and compared with values obtained in our study.

4 Experimental Comparison

Trials were performed to measure the predictive accuracy and standard deviation of algorithms on the different datasets mentioned in Table 1 above. Classification accuracy is measured by splitting the data into a training and a test (or hold-out) set , presenting the algorithm with the training set to induce a concept description and then measuring the percentage of correct predictions made by that concept description on the test set. In each trial, 80% of the training examples were selected at random from the entire dataset and the remaining 20% of the data used for testing. After splitting, data are converted into a suitable format required for each algorithm. Five such trials were performed on different splits on training and test for each of the domains and results averaged. In each experiment, the algorithms being compared were trained (and tested) on identical data to ensure that differences in performance were due entirely to the classification performance of algorithms.

4.1 Learning Parameters

In the training phase, some algorithms require the user to specify learning parameters while others do not. For those algorithms requiring special learning parameters, default or suggested values were used. Details of parameter specification for each algorithm are described as follows:

1. *C4.5 and C4.5 rules:* no special learning parameters. Default values were used.
2. *ITI:* leave-one-out cross validation was used to validate the induced tree. Integer-valued attributes, normally treated as continuous values by ITI, are encoded symbolically to avoid the effects of: "*If number of legs ≤ 3.5 then ...*".
3. *LMDT:* does not require any special learning parameters but requires data for validating the resulting tree. Therefore, the training set was further randomly split into training and validating sets (80% /20% accordingly). Running LMDT in this experiment includes identification training, validating and testing data files.

higher averaged accuracy over five runs. However, the approach is more suited to comparing two algorithms, and becomes overly complicated as the number of algorithms in the comparison increases.

In this comparison we present both methods to compare classification accuracy and standard deviation. The resultant classification accuracy measurement and statistical analysis is presented in a later section of this paper.

4.3 Classification Averages and Standard Deviations

This section presents results of the comparative analysis in a number of dimensions: average classification accuracy and standard deviations, types of attribute values, data derivations, and dataset size.

Raw results of the experiment are presented in Tables 2 and 3. In these tables, each row corresponds to a different data domain connecting the observed accuracy and standard deviation of one algorithm with another. The highest classification accuracy and lowest standard deviations in each row are highlighted in bold-face, "×" indicates that the algorithm cannot run on this data.

4.4 Algorithm Comparison: All Types of Data

Tables 6 and 7 show the percentage accuracies and standard deviation of five algorithms (C4.5, C4.5 rules, ITI, LMDT, and CN2) over the classification tasks. The highest and lowest value classification accuracies and standard deviations in each line are highlighted in bold-face.

The values of average classification accuracies and standard deviations of these five algorithms on the first 28 tasks are provided in Table 8. Observe that LMDT dominates with respect to the mean classification accuracies. From the first 27 tasks, LMDT gives maximum values on 22 tasks while C4.5, C4.5 rules, CN2, and ITI give maximum values on only two or three tasks. The results of comparison of mean accuracies using paired, two-tailed *t*-test on the above data are presented in Table 9.

4.5 Comparisons of Standard Deviation

LMDT gives the best averaged standard deviation over 28 tasks (>99% significant). Its average standard deviation is 8.15 while values for C4.5, C4.5 rules, CN2, and ITI are only 4.88, 4.74, 4.84 and

5.02 respectively. Statistical comparison between the four other algorithms over these 28 tasks shows no significant differences wrt to classification accuracy or standard deviation. Note that C4.5 rules gives both slightly higher average accuracy and lower standard deviation than other algorithms.

4.6 Algorithm Comparison: Numeric Data

A selection of datasets containing only numeric predictive feature values, allows us to examine the performance of 8 different algorithms on 17 numeric datasets in total. The resulting classification accuracies and standard deviations are presented in Tables 10 and 11. The values of classification accuracy averaged over all datasets are given in the bottom of each table. Again, bold-face type is used to highlight the best accuracy and standard deviation on each line of the tables.

As can be seen from the tables, further inclusion of three more algorithms in the comparison (LVQ, OC1, and Nevprop) does not affect the domination of LMDT wrt classification accuracy. Over the 17 tasks presented in the tables, LMDT gives the maximum and OC1 minimum mean values. Furthermore, statistical analysis supports the intuitive observation that LMDT performs (in general) well and OC1 (in general) poorly. The results of statistical comparison between LMDT and OC1 with other algorithms are presented in Tables 12 and 13. OC1 gives the highest value averaged over 17 tasks with respect to standard deviation. The lowest standard deviation is produced by C4.5 and C4.5 rules but statistically significant differences are found only between those algorithms and OC1 (>98% significant) and LMDT (>94% significant). There are no significant differences in either classification accuracy and standard deviation between C4.5, C4.5 rules, ITI, CN2, LVQ and Nevprop classifiers. C4.5 rules gives noticeably higher average accuracy than C4.5, ITI, LVQ, OC1, Nevprop and lower average standard deviation than all except C4.5.

4.7 Algorithm Comparison: Numeric Data without Missing Values

Selecting just those datasets containing no missing values allows the opportunity to compare the performance of 11 different algorithms[12] on 13 datasets. Tables 4 and 5 present results of average classification accuracies and standard deviations measured in the experiment.

As illustrated in Tables 4 and 5, LMDT produces higher classification accuracies. Results of two-tailed *t*-tests show significant differences between LMDT's average accuracy and C4.5 (>96%), LVQ (>99%), OC1 (>99%), K5 (>95%), Q* (>97%). There are no significant differences in average accuracy over 13 tasks between LMDT, CN2, ITI, and C4.5 rules and no significant difference can be found between CN2, ITI and C4.5. ITI gives a lower standard deviation than LVQ (>94% significant), OC1 (>98% significant), K5 (>94% significant), Q* (>95% significant). Both C4.5 algorithms give significantly lower standard deviations than OC1, K5 and Q* (>96% significant). There are no significant differences in standard deviation between any of the other algorithms.

[12] C4.5, C4.5 rules, ITI, LMDT, CN2, LVQ, OC1, Nevprop, K5, Q*, and RBF.

#	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}	A_{11}	A_{12}
1	92.19	93.97	98.47	**99.05**	86.60	×	×	×	×	×	×	×
2	77.05	74.84	73.66	**81.71**	69.53	×	×	×	×	×	×	×
3	85.91	85.66	83.47	**88.93**	81.66	×	×	×	×	×	×	×
4	78.42	81.23	75.93	**84.59**	83.43	×	×	×	×	×	×	×
5	70.51	70.51	68.44	**75.36**	68.78	×	×	×	×	×	×	×
6	99.41	**99.43**	99.36	87.90	98.24	×	×	×	×	×	×	×
7	84.16	85.83	86.65	85.83	**86.66**	×	×	×	×	×	×	×
8	62.57	60.05	59.48	**66.88**	57.91	×	×	×	×	×	×	×
9	89.91	92.14	**94.20**	87.97	91.37	×	×	×	×	×	×	×
10	**100.0**	99.96	**100.0**	**100.0**	**100.0**	93.46	×	×	×	×	×	×
11	71.15	67.07	62.23	71.20	66.33	**72.62**	63.05	×	×	×	×	×
12	96.28	94.66	96.03	**97.20**	95.92	95.45	96.15	94.56	×	×	×	×
13	74.19	76.13	77.40	**83.45**	80.32	60.79	72.26	76.31	×	×	×	×
14	41.69	37.24	37.67	**49.20**	42.47	40.37	22.73	1.46	×	×	×	×
15	70.23	67.96	67.49	60.59	70.23	60.69	57.72	44.08	69.09	**74.78**	69.54	×
16	94.25	94.68	91.14	**95.74**	94.39	94.82	93.24	95.05	96.38	95.46	94.89	×
17	96.49	96.36	96.03	**97.61**	90.56	89.39	93.33	95.21	91.24	95.92	90.03	×
18	91.56	91.82	**93.65**	86.89	90.98	88.58	88.29	83.80	85.91	89.70	87.60	×
19	40.17	39.84	38.47	55.49	37.17	55.71	54.28	33.12	**68.54**	60.00	65.70	×
20	91.09	91.90	91.09	**95.40**	91.09	68.90	87.31	95.41	69.49	74.35	67.87	×
21	71.02	71.55	73.16	**73.51**	72.19	71.28	50.00	68.52	71.37	68.50	70.57	×
22	65.14	65.39	63.00	71.54	64.31	64.13	65.57	**77.72**	66.43	61.43	59.85	×
23	83.52	**99.17**	92.89	89.61	98.18	65.61	78.56	96.91	84.32	65.70	72.19	96.97
24	64.61	75.01	76.76	93.27	80.89	89.54	92.50	91.04	83.96	69.21	89.06	**97.04**
25	91.60	91.58	91.25	**95.45**	91.92	92.55	93.89	90.34	91.94	92.10	85.64	93.55
26	70.03	75.95	78.63	78.94	75.21	52.26	63.73	78.91	55.21	74.84	70.03	**79.65**
27	74.74	77.75	74.81	**84.34**	78.06	80.00	39.92	72.47	75.41	74.91	×	77.96
28	90.27	90.00	90.93	**96.61**	91.91	91.42	66.68	92.86	67.64	74.94	×	93.23
29	**86.85**	×	×	×	70.64	79.40	×	×	×	×	×	×

Table 2: Classification accuracies of experimental algorithms, averaged over five runs.

Note: **Columns**: A_1 =C4.5, A_2=C4.5 rules, A_3=ITI, A_4=LMDT, A_5=CN2, A_6 =LVQ, A_7=OC1, A_8=Nevprop, A_9 =K5, A_{10} = Q*, A_{11}=RBF, A_{12}= SNNS.

Rows: 1=Annealing, 2=Audiology, 3=Credit Screening, 4=Hepatitis, 5=Breast Cancer, 6=Hypothyroid, 7=Labor Relation, 8=Post Operative, 9=Soybean, 10=Mushroom, 11=Echocardiogram, 12=Voting, 13=Cleveland, 14=Primary Tumor, 15=Glass, 16=Breast Cancer Wincosin, 17=Image, 18=Ionosphere, 19=Lung Cancer, 20=Wine, 21=Pima Indian, 22=Bupa, 23=Tic Tac Toe, 24=Balance Scale, 25=Iris, 26=Hayes-Roth, 27=Lymphography, 28=Zoo, 29=Letter Recognition.

#	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}	A_{11}	A_{12}
1	1.54	2.18	1.14	**1.08**	3.32	×	×	×	×	×	×	×
2	**4.77**	6.23	6.16	13.28	5.03	×	×	×	×	×	×	×
3	**2.25**	2.55	3.23	4.60	2.49	×	×	×	×	×	×	×
4	6.66	**3.31**	6.76	8.14	8.85	×	×	×	×	×	×	×
5	5.36	6.58	**5.02**	9.15	6.32	×	×	×	×	×	×	×
6	0.24	**0.17**	0.28	23.59	0.68	×	×	×	×	×	×	×
7	7.30	5.65	**4.35**	7.92	8.97	×	×	×	×	×	×	×
8	10.65	12.47	**9.30**	17.58	9.30	×	×	×	×	×	×	×
9	4.15	2.94	**1.74**	2.41	2.41	×	×	×	×	×	×	×
10	**0.00**	0.13	**0.00**	**0.00**	**0.00**	2.63	×	×	×	×	×	×
11	9.38	**5.07**	11.29	12.67	5.90	7.44	12.23	×	×	×	×	×
12	1.70	2.09	1.95	1.53	2.27	2.48	**1.93**	2.50	×	×	×	×
13	3.79	4.34	4.30	6.49	3.80	5.05	4.61	**3.63**	×	×	×	×
14	7.95	4.84	5.76	15.49	4.49	4.15	7.14	**1.37**	×	×	×	×
15	7.23	**6.28**	7.96	11.25	8.34	10.24	9.10	6.29	7.81	6.98	7.35	×
16	2.36	2.40	6.79	3.03	3.60	4.91	3.60	2.20	2.31	**1.94**	2.83	×
17	**1.09**	1.25	1.46	1.42	1.81	2.09	2.72	1.53	6.23	1.11	1.81	×
18	2.82	2.58	2.71	3.51	3.29	3.36	**2.21**	3.81	4.14	4.70	6.45	×
19	14.20	18.92	13.52	32.20	13.79	12.48	17.53	14.83	**11.96**	18.60	16.27	×
20	5.84	5.09	6.24	5.22	6.11	4.84	8.45	**2.22**	6.86	6.64	5.16	×
21	**2.10**	3.92	2.16	4.30	2.36	4.46	22.40	3.19	3.67	8.19	2.39	×
22	5.74	6.05	**4.23**	6.63	7.99	7.14	8.45	11.97	7.22	4.25	7.92	×
23	2.44	1.05	2.38	8.79	0.95	2.99	5.88	1.32	2.70	3.16	3.35	**0.93**
24	3.35	3.98	3.00	2.95	3.38	4.39	2.07	7.12	7.53	19.09	2.38	**1.13**
25	5.09	5.09	4.81	4.71	5.95	**3.73**	4.68	7.45	4.10	5.28	27.37	4.04
26	7.69	7.24	**5.96**	8.69	7.20	9.27	12.94	11.84	10.85	8.69	8.62	6.10
27	3.23	**3.20**	6.92	9.94	6.05	7.73	21.08	12.93	4.97	6.50		7.74
28	7.59	7.24	6.11	**1.56**	5.95	6.26	30.36	4.62	20.03	23.80		5.13
29	0.62	×	×	×	**0.34**	4.70	×	×	×	×	×	×

Table 3: Standard deviation of experimental algorithms.

Note: **Columns**: A_1 =C4.5, A_2=C4.5 rules, A_3=ITI, A_4=LMDT, A_5=CN2, A_6 =LVQ, A_7=OC1, A_8=Nevprop, A_9 =K5, A_{10} = Q*, A_{11}=RBF, A_{12}= SNNS.

Rows: 1=Annealing, 2=Audiology, 3=Credit Screening, 4=Hepatitis, 5=Breast Cancer, 6=Hypothyroid, 7=Labor Relation, 8=Post Operative, 9=Soybean, 10=Mushroom, 11=Echocardiogram, 12=Voting, 13=Cleveland, 14=Primary Tumor, 15=Glass, 16=Breast Cancer Wincosin, 17=Image, 18=Ionosphere, 19=Lung Cancer, 20=Wine, 21=Pima Indian, 22=Bupa, 23=Tic Tac Toe, 24=Balance Scale, 25=Iris, 26=Hayes-Roth, 27=Lymphography, 28=Zoo, 29=Letter Recognition.

#	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}	A_{11}
1	70.23	67.96	67.49	60.59	70.23	60.69	57.72	44.08	69.09	**74.78**	69.54
2	96.49	96.36	96.03	**97.61**	90.56	89.39	93.33	95.21	91.24	95.92	90.03
3	91.56	91.82	**93.65**	86.89	90.98	88.58	88.29	83.80	85.91	89.70	87.60
4	40.17	39.84	38.47	55.49	37.17	55.71	54.28	33.12	**68.54**	60.00	65.70
5	91.09	91.90	91.09	**95.40**	91.09	68.90	87.31	95.41	69.49	74.35	67.87
6	71.02	71.55	73.16	**73.51**	72.19	71.28	50.00	68.52	71.37	68.50	70.57
7	65.14	65.39	63.00	71.54	64.31	64.13	65.57	**77.72**	66.43	61.43	59.85
8	83.52	**99.17**	92.89	89.61	98.18	65.61	78.56	96.91	84.32	65.70	72.19
9	64.61	75.01	76.76	**93.27**	80.89	89.54	92.50	91.04	83.96	69.21	89.06
10	91.60	91.58	91.25	**95.45**	91.92	92.55	93.89	90.34	91.94	92.10	85.64
11	70.03	75.95	78.63	**78.94**	75.21	52.26	63.73	78.91	55.21	74.84	70.03
12	74.74	77.75	74.81	**84.34**	78.06	80.00	39.92	72.47	75.41	74.91	×
13	90.27	90.00	90.93	**96.61**	91.91	91.42	66.68	92.86	67.64	74.94	×
Avr	76.96	79.56	79.09	**83.02**	79.44	74.62	71.68	78.49	75.43	75.11	75.28

Table 4: Average classification accuracies of algorithms, running on datasets with numerical attributes without missing values.

Note: **Columns**: A_1 =C4.5, A_2=C4.5 rules, A_3=ITI, A_4=LMDT, A_5=CN2, A_6 =LVQ, A_7=OC1, A_8=Nevprop, A_9 =K5, A_{10} = Q*, A_{11}=RBF.

Rows: 1=Glass, 2=Image, 3=Ionosphere, 4=Lung Cancer, 5=Wine, 6=Pima Indian, 7=Bupa, 8=Tic Tac Toe, 9=Balance Scale, 10=Iris, 11= Hayes-Roth, 12= Lymphography, 13=Zoo, Avr=Average.

#	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}	A_{11}
1	7.23	**6.28**	7.96	11.25	8.34	10.24	9.10	6.29	7.81	6.98	7.35
2	**1.09**	1.25	1.46	1.42	1.81	2.09	2.72	1.53	6.23	1.11	1.81
3	2.82	2.58	2.71	3.51	3.29	3.36	**2.21**	3.81	4.14	4.70	6.45
4	14.20	18.92	13.52	32.20	13.79	12.48	17.53	14.83	**11.96**	18.60	16.27
5	5.84	5.09	6.24	5.22	6.11	4.84	8.45	**2.22**	6.86	6.64	5.16
6	**2.10**	3.92	2.16	4.30	2.36	4.46	22.40	3.19	3.67	8.19	2.39
7	5.74	6.05	**4.23**	6.63	7.99	7.14	8.45	11.97	7.22	4.25	7.92
8	2.44	1.05	2.38	8.79	**0.95**	2.99	5.88	1.32	2.70	3.16	3.35
9	3.35	3.98	3.00	2.95	3.38	4.39	2.07	7.12	7.53	19.09	**2.38**
10	5.09	5.09	4.81	4.71	5.95	**3.73**	4.68	7.45	4.10	5.28	27.37
11	7.69	7.24	**5.96**	8.69	7.20	9.27	12.94	11.84	10.85	8.69	8.62
12	3.23	**3.20**	6.92	9.94	6.05	7.73	21.08	12.93	4.97	6.50	×
13	7.59	7.24	6.11	**1.56**	5.95	6.26	30.36	4.62	20.03	23.80	×
Avr	5.26	5.53	**5.19**	7.78	5.63	6.07	11.37	6.86	7.54	9.00	8.10

Table 5: Standard deviation of algorithms running on datasets with numerical attributes without missing values.

Note: **Columns**: A_1 =C4.5, A_2=C4.5 rules, A_3=ITI, A_4=LMDT, A_5=CN2, A_6 =LVQ, A_7=OC1, A_8=Nevprop, A_9 =K5, A_{10} = Q*, A_{11}=RBF.

Rows: 1=Glass, 2=Image, 3=Ionosphere, 4=Lung Cancer, 5=Wine, 6=Pima Indian, 7=Bupa, 8=Tic Tac Toe, 9=Balance Scale, 10=Iris, 11= Hayes-Roth, 12= Lymphography, 13=Zoo, Avr=Average

#	Database	C4.5	C4.5 rules	ITI	LMDT	CN2
1	Annealing	92.19	93.97	98.47	**99.05**	86.60
2	Audiology	77.05	74.84	73.66	**81.71**	69.53
3	Cred. Screening	85.91	85.66	83.47	**88.93**	81.66
4	Hepatitis	78.42	81.23	75.93	**84.59**	83.43
5	Breast Cancer	70.51	70.51	68.44	**75.36**	68.78
6	Hypothyroid	99.41	**99.43**	99.36	87.90	98.24
7	Labor Relation	84.16	85.83	86.65	85.83	**86.66**
8	Post Operative	62.57	60.05	59.48	**66.88**	57.91
9	Soybean	89.91	92.14	**94.20**	87.97	91.37
10	Mushroom	**100.0**	99.96	**100.0**	**100.0**	**100.0**
11	Echocardiogram	71.15	67.07	62.23	**71.20**	66.33
12	Voting	96.28	94.66	96.03	**97.20**	95.92
13	Cleveland	74.19	76.13	77.40	**83.45**	80.32
14	Primary Tumor	41.69	37.24	37.67	**49.20**	42.47
15	Glass	**70.23**	67.96	67.49	60.59	70.23
16	Breast Cancer W.	94.25	94.68	91.14	**95.74**	94.39
17	Image	96.49	96.36	96.03	**97.61**	90.56
18	Ionosphere	91.56	91.82	**93.65**	86.89	90.98
19	Lung Cancer	40.17	39.84	38.47	**55.49**	37.17
20	Wine	91.09	91.90	91.09	**95.40**	91.09
21	Pima Indian	71.02	71.55	73.16	**73.51**	72.19
22	Bupa	65.14	65.39	63.00	**71.54**	64.31
23	Tic Tac Toe	83.52	**99.17**	92.89	89.61	98.18
24	Balance Scale	64.61	75.01	76.76	**93.27**	80.89
25	Iris	91.60	91.58	91.25	**95.45**	91.92
26	Hayes-Roth	70.03	75.95	78.63	**78.94**	75.21
27	Lymphography	74.74	77.75	74.81	**84.34**	78.06
28	Zoo	90.27	90.00	90.93	**96.61**	91.91

Table 6: Percentage accuracies of five algorithms: C4.5, C4.5 rules, ITI, LMDT, and CN2.

#	Database	C4.5	C4.5 rules	ITI	LMDT	CN2
1	Annealing	1.54	2.18	1.14	**1.08**	3.32
2	Audiology	**4.77**	6.23	6.16	13.28	5.03
3	Cred. Screening	**2.25**	2.55	3.23	4.60	2.49
4	Hepatitis	6.66	**3.31**	6.76	8.14	8.85
5	Breast Cancer	5.36	6.58	**5.02**	9.15	6.32
6	Hypothyroid	0.24	**0.17**	0.28	23.59	0.68
7	Labor Relation	7.30	5.65	**4.35**	7.92	8.97
8	Post Operative	10.65	12.47	**9.30**	17.58	**9.30**
9	Soybean	4.15	2.94	**1.74**	2.41	2.41
10	Mushroom	**0.00**	0.13	**0.00**	**0.00**	**0.00**
11	Echocardiogram	9.38	**5.07**	11.29	12.67	5.90
12	Voting	**1.70**	2.09	1.95	1.53	2.27
13	Cleveland	**3.79**	4.34	4.30	6.49	3.80
14	Primary Tumor	7.95	4.84	5.76	15.49	**4.49**
15	Glass	7.23	**6.28**	7.96	11.25	8.34
16	Br. Cancer W.	**2.36**	2.40	6.79	3.03	3.60
17	Image	**1.09**	1.25	1.46	1.42	1.81
18	Ionosphere	2.82	**2.58**	2.71	3.51	3.29
19	Lung Cancer	14.20	18.92	**13.52**	32.20	13.79
20	Wine	5.84	**5.09**	6.24	5.22	6.11
21	Pima Indian	**2.10**	3.92	2.16	4.30	2.36
22	Bupa	5.74	6.05	**4.23**	6.63	7.99
23	Tic Tac Toe	2.44	1.05	2.38	8.79	**0.95**
24	Balance Scale	3.35	3.98	3.00	**2.95**	3.38
25	Iris	5.09	5.09	4.81	**4.71**	5.95
26	Hayes-Roth	7.69	7.24	**5.96**	8.69	7.20
27	Lymphography	3.23	**3.20**	6.92	9.94	6.05
28	Zoo	7.59	7.24	6.11	**1.56**	5.95

Table 7: Standard deviation of five algorithms: C4.5, C4.5 rules, ITI, LMDT, and CN2.

Feature	C4.5	C4.5 rules	ITI	LMDT	CN2
Average accuracy on tasks 1-28	79.22	80.27	79.72	**83.37**	79.87
Standard deviation on tasks 1-28	4.88	**4.74**	4.84	8.15	5.02

Table 8: Average classification accuracies and average standard deviations of five algorithms: C4.5, C4.5 rules, ITI, LMDT, and CN2.

Algorithms	Difference (Mean X-Y)	Significance
LMDT - C4.5	+4.15	√
LMDT - C4.5 rules	+3.10	√
LMDT - ITI	+3.65	√
LMDT - CN2	+3.50	√

Table 9: Comparison of mean accuracies using paired, two-tailed t-test between LMDT and the four algorithms: C4.5, C4.5 rules, ITI, and CN2.

#	Database	C4.5	C4.5 r.	ITI	LMDT	CN2	LVQ	OC1	Nevpr
1	Voting	96.28	94.66	96.03	**97.20**	95.92	95.45	96.15	94.56
2	Primary Tumor	41.69	37.24	37.67	**49.20**	42.47	40.37	22.73	1.46
3	Cleveland	74.19	76.13	77.40	**83.45**	80.32	60.79	72.26	76.31
4	Lung Cancer	40.17	39.84	38.47	55.49	37.17	**55.71**	54.28	33.12
5	Bupa	65.14	65.39	63.00	71.54	64.31	64.13	65.57	**77.72**
6	Glass	**70.23**	67.96	67.49	60.59	**70.23**	60.69	57.72	44.08
7	Image	96.49	96.36	96.03	**97.61**	90.56	89.39	93.33	95.21
8	Ionosphere	91.56	91.82	**93.65**	86.89	90.98	88.58	88.29	83.80
9	Pima Indian	71.02	71.55	73.16	**73.51**	72.19	71.28	50.00	68.52
10	Wine	91.09	91.90	91.09	**95.40**	91.09	68.90	87.31	95.41
11	Breast Cancer W.	94.25	94.68	91.14	**95.74**	94.39	94.82	93.24	95.05
12	Tic Tac Toe	83.52	**99.17**	92.89	89.61	98.18	65.61	78.56	96.91
13	Hayes-Roth	70.03	75.95	78.63	**78.94**	75.21	52.26	63.73	78.91
14	Balance Scale	64.61	75.01	76.76	**93.27**	80.89	89.54	92.50	91.04
15	Iris	91.60	91.58	91.25	**95.45**	91.92	92.55	93.89	90.34
16	Zoo	90.27	90.00	90.93	**96.61**	91.91	91.42	66.68	92.86
17	Lymphography	74.74	77.75	74.81	**84.34**	78.06	80.00	39.92	72.47
	Average	76.88	78.65	78.26	**82.64**	79.16	74.21	71.54	75.75

Table 10: Average classification accuracies of 7 algorithms C4.5, C4.5 rules, ITI, LMDT, CN2, LVQ, and OC1 on numeric data. The best classification accuracies are emboldened

#	Database	C4.5	C4.5 r.	ITI	LMDT	CN2	LVQ	OC1	Nevpr
1	Voting	1.70	2.09	1.95	**1.53**	2.27	2.48	1.93	2.50
2	Primary Tumor	7.95	4.84	5.76	15.49	4.49	4.15	7.14	**1.37**
3	Cleveland	3.79	4.34	4.30	6.49	3.80	5.05	4.61	**3.63**
4	Lung Cancer	14.20	18.92	13.52	32.20	13.79	**12.48**	17.53	14.83
5	Bupa	5.74	6.05	**4.23**	6.63	7.99	7.14	8.45	11.97
6	Glass	7.23	**6.28**	7.96	11.25	8.34	10.24	9.10	6.29
7	Image	**1.09**	1.25	1.46	1.42	1.81	2.09	2.72	1.53
8	Ionosphere	2.82	2.58	2.71	3.51	3.29	3.36	**2.21**	3.81
9	Pima Indian	**2.10**	3.92	2.16	4.30	2.36	4.46	22.40	3.19
10	Wine	5.84	5.09	6.24	5.22	6.11	4.84	8.45	**2.22**
11	Breast Cancer W.	2.36	2.40	6.79	3.03	3.60	4.91	3.60	**2.20**
12	Tic Tac Toe	2.44	1.05	2.38	8.79	**0.95**	2.99	5.88	1.32
13	Hayes-Roth	7.69	7.24	**5.96**	8.69	7.20	9.27	12.94	11.84
14	Balance Scale	3.35	3.98	3.00	2.95	3.38	4.39	**2.07**	7.12
15	Iris	5.09	5.09	4.81	4.71	5.95	**3.73**	4.68	7.45
16	Zoo	7.59	7.24	6.11	**1.56**	5.95	6.26	30.36	4.62
17	Lymphography	3.23	3.20	6.92	9.94	6.05	7.73	21.08	12.93
	Average	**4.95**	5.03	5.07	7.51	5.14	5.62	9.71	5.81

Table 11: Standard deviations of 7 algorithms C4.5, C4.5 rules, ITI, LMDT, CN2, LVQ, and OC1 on numeric data. Lowest values are in bold.

Algorithms	Difference (Mean X-Y)	Significance
LMDT - C4.5	+5.76	√
LMDT - C4.5 rules	+3.99	√
LMDT - ITI	+4.38	√
LMDT - CN2	+3.48	√
LMDT - LVQ	+8.43	√
LMDT - OC1	+11.1	√
LMDT - Nevprop	+6.89	√

Table 12: Results of two-tailed *t*-test comparison between LMDT and other algorithms on numeric data. LMDT performance is significantly better for each pair.

Algorithms	δ (Mean X-Y)	Significance
OC1 - C4.5	-5.34	√
OC1 - C4.5 rules	-7.10	√
OC1 - ITI	-6.72	√
OC1 - LMDT	-11.1	√
OC1 - CN2	-7.62	√
OC1 - LVQ	-2.67	√
OC1 - Nevprop	-4.21	√

Table 13: Results of two-tailed *t*-test comparison between OC1 and other algorithms on numeric data. OC1's performance is significantly worse in the majority of comparison pairs.

The selection of 6 datasets: Tic-Tac-Toe, Balance Scale, Iris, Hayes-Roth, Lymphography, and Zoo, allows us to add one more algorithm into the comparison: SNNS. Because of the lack of built-in data normalization facilities, SNNS can only be applied to datasets with integer-valued attributes encoded by the binary encoding technique mentioned in Section 4.1 previously (or datasets with real-valued attributes on the same scale or normalized into mean zero values). The measured accuracies and standard deviation of SNNS on these datasets is 96.97, 97.04, 93.55, 76.95, 77.96, 92.23 and 0.93, 1.13, 4.04, 6.10, 7.14, 5.13 respectively. Averaged over 6 datasets, SNNS gives the highest classification accuracy of 89.73%, this figure is higher (but the difference is not statistically significant) than LMDT (89.70%). On this set of data the standard deviation is 4.18, lower (but again not significantly) than LMDT (6.11). The average classification accuracy produced by SNNS is significantly higher than those produced by K5, Q*, Nevprop and OC1 algorithms (>94 % significant). No significant differences in standard deviation can be found between any other of the studied algorithms.

4.8 Algorithm Comparison: Symbolic or Mixture Symbolic and Numeric Data

The result of the comparison between five algorithms (C4.5, C4.5 rules, ITI, LMDT, and CN2) on 9 datasets (Annealing, Audiology, Breast Cancer, Credit Screening, Hepatitis, Hypothyroid, Labor Relation, Post Operative and Soybean) that consist of symbolic, or mixed symbolic and numeric predictive feature values, shows that although LMDT produces the highest accuracy, there are no significant differences between its accuracy and the performance of any of the other algorithms studied. However, C4.5 rules produces significantly (>94% significant) higher average classification accuracy (82.63%) than CN2 (80.46%) in these domains. LMDT again produces the highest standard deviation (10.67) and significantly higher (>95% significant) than ITI (4.53).

4.8.1 Algorithm Comparison: Medical and Biological Domains

Selective analysis of classification accuracies and standard deviations on 12 datasets with a medical, bio-medical or natural science derivation[13] (these mostly store data on disease diagnosis) shows LMDT produces significantly better mean accuracy in these domains. On all the 12 tasks mentioned above, LMDT gives 75.06% mean accuracy, better than C4.5 (71.85%, >96% significant), C4.5 rules (71.79%, >97% significant), ITI (70.18%, >99% significant), and CN2 (71.90%, >96% significant). On the last 7 datasets, starting from Echocardiogram, the average accuracy produced by LMDT is 73.29%, higher (>95% significant) than the values produced by C4.5, C4.5 rules, ITI, CN2, LVQ, OC1 and Nevprop (65.9%, 65.44%, 63.53%, 66.15%, 66.92%, 58.72% and 59.35% respectively). Here again, we found that C4.5 rules gives

[13] Soybean, Hypothyroid, Hepatitis, Post Operative, Breast Cancer, Echocardiogram, Cleveland, Primary Tumor, Bupa, Breast Cancer W., Lymphography and Lung Cancer.

the lowest average standard deviation (5.86 on all 12 datasets and 6.4 on the last 7 datasets) and lower (>98% significant) than LMDT (13.37 on all 12 sets and 12.35 on the last 7 sets).

The addition of four more datasets with biological derivation[14] does not alter the observations made above. LMDT continues to produce significantly higher (>98% significant) average classification accuracies than all other algorithms (C4.5 rules, ITI, CN2, LVQ, and OC1) on all 16 tasks. No significant differences can be found within this group of six algorithms with respect to average accuracies. On all 16 tasks, all four algorithms: CN2, ITI, C4.5 rules and C4.5 give lower standard deviations than LMDT (>95% significant) while there is no significant difference in standard deviations within this group of four.

4.8.2 Algorithm Comparison: the Size Dimension

Analysis of average classification accuracies and standard deviations along the size of datasets shows LMDT produces better average accuracies on any size dataset compared to all other algorithms. On 13 datasets (Labor Relations, Post Operative, Audiology, Breast cancer, Hepatitis, Echocardiogram, Glass, Wine, Lung cancer, Iris, Hayes-Roth, Zoo and Lymphography) where the number of cases is less than 300, the average accuracy produced by LMDT is 79.41%, higher than C4.5 (74.77%, >99% significant), C4.5 rules (74.96%, >99% significant), ITI (73.77%, >99.99% significant), and CN2 (74.48%, >98% significant). There are no significant differences in average classification accuracies between C4.5, C4.5 rules, ITI and CN2. On the last 7 datasets listed above (starting with Glass), LMDT produces average classification accuracy higher than Nevprop (>96% significant) and OC1 (>94% significant). On the last 4 datasets (Iris, Hayes-Roth, Zoo, and Lymphography), LMDT again produces higher average classification accuracies (>99% significant) than C4.5, C4.5 rules, and CN2. On this set of data, SNNS's average accuracy is higher than ITI (>98% significant) and C4.5 rules (>94% significant). However, no differences in average accuracies could be found between LMDT and ITI, LVQ, OC1, Nevprop, and TOOLDIAG classifiers as well as between SNNS and any other algorithm.

Further extension of database size to 600 and 900 cases does not change the classication dominance of LMDT. When four more datasets: Voting, Primary Tumor, Bupa and Ionosphere are included, LMDT continues to produce significantly higher average accuracy than C4.5, C4.5 rules, ITI, LVQ, OC1 and Nevprop. For datasets with the number of cases >900 (Hypothyroid, Mushroom, Tic-Tac-Toe, Image), C4.5 rules produces the highest average accuracy (but the differences are not significant). Only three algorithms (C4.5, LVQ and CN2) are able to run on large data (Letter Recognition). On this data C4.5 produces the highest classification accuracy (86.85%), while CN2 and LVQ yield only 70.64% and 79.40% respectively.

4.8.3 Summary of Performance Comparison

The analysis of average classification accuracies leads to some interesting observations:

All Data Among those algorithms that can run on all types of data, LMDT produces the highest average classification accuracy and standard deviation. LMDT, averaged over all 27 datasets on which it can be applied, is significantly better than C4.5, C4.5 rules, ITI and CN2. However, LMDT's average standard deviation is significantly higher than the above mentioned algorithms. There are no statistically significant differences in standard deviation and average classification accuracies between C4.5, C4.5 rules, ITI, and CN2.

All Numeric Data In comparison between 8 algorithms (LMDT, C4.5, C4.5 rules, ITI, CN2, LVQ, OC1, and Nevprop) running on numeric-valued datasets, LMDT again produces significant the highest classification accuracy (and OC1 the lowest), averaged over 17 datasets. OC1 also produces the highest average standard deviation. C4.5 rules produces higher average accuracy than C4.5, ITI, LVQ, OC1 and Nevprop.

No Missing values in Numeric data A selection of numeric datasets without missing values includes the 3 TOOLDIAG classifiers into the comparison. LMDT gives significantly higher average classification accuracy than C4.5, LVQ, OC1, and TOOLDIAG classifiers. Both C4.5 and C4.5 rules produce significantly lower average classification accuracy than OC1 and TOOLDIAG classifiers.

Mixed Symbolic and Numeric Data On 9 datasets with symbolic or mixed symbolic and numeric-valued attribute values, C4.5 rules produces significantly higher average accuracy than CN2. LMDT continues to produce the highest average classification accuracy and standard deviation, although the differences are not significant.

Epidemiological/Clinical Data Sources The result of average accuracy analysis on datasets derived from epidemiological health-care sources (those with a bio-medical nature) shows LMDT dominates with respect to average classification accuracy while C4.5 and C4.5 rules dominate with respect to average standard deviation.

Large Datasets On 4 large datasets (where the number of examples exceeds 900 cases) C4.5 rules produces the highest average accuracy. Only C4.5, CN2 and LVQ can run on the Letter Recognition database (20,000 cases) under the conditions of this experiment. From these three algorithms, C4.5 gives the highest classification accuracy.

[14] Mushroom, Wine, Iris, and Zoo.

(Group 1/Group 2)	C4.5	ITI	C4.5r.	CN2	Nevprop	OC1	LVQ	K5	Q*	RBF
LMDT	8/1	9/1	7/1	7/1	9/1	9/0	9/0	6/1	7/2	6/1
SNNS	3/0	2/1	2/1	2/1	2/1	5/1	3/1	4/0	3/0	3/0

Table 14: Comparison between algorithms of Group 1 and others in the cardinality of datasets in which the classification accuracy of one algorithm is significantly higher than another.

Group 2/Group 3	OC1	LVQ	K5	Q*	RBF
C4.5	8/2	9/3	5/2	3/1	4/1
C4.5 rules	12/1	8/3	6/2	2/1	5/2
ITI	10/3	8/3	6/3	3/2	5/1
CN2	9/3	6/4	5/1	4/2	4/2
Nevprop	8/3	6/4	6/3	5/2	5/2

Table 15: Comparison between algorithms of Group 2 and Group 3 in the cardinality of datasets on which the classification accuracy of one algorithm is significantly higher than another.

4.9 Cardinality Comparison

In this section, the second comparative approach, the so called *Cardinality Comparison* described in Section 4.2 is applied. The results of comparing the number of datasets, on which one particular algorithm produces higher average classification accuracy, are presented in Figure 8, Tables 14 and 15.

The 12 experimental algorithms of this study can be divided into 3 groups according to the relative average classification accuracy over 10 runs. The first we call *Group 1*. Group 1 contains only 2 algorithms: LMDT and SNNS. The second, *Group 2* consists of 5 algorithms: C4.5, C4.5 rules, CN2, ITI, and Nevprop. The last, *Group 3* contains LVQ, OC1 and 3 TOOLDIAG classifies: K5, Q* and RBF. Within each group there is no statistical significance between the algorithms in terms of average classification performance.

However, algorithms in Group 1 produce statistical significantly higher classification accuracy on more datasets than algorithms in Group 2 and Group 3. The algorithms of the Group 2, in turn, produce statistically significant higher classification accuracy on more datasets than all algorithms of the Group 3.

According to the results of the statistical analysis, LMDT and SNNS produce higher classification accuracy on more datasets than all other algorithms. However, because the number of datasets on which SNNS can run is limited, its inclusion in Group 1 is reluctant. The result of the comparison is represented in Table 14. The method of representing data is as follows: each entry of the table contains a fraction. The numerator is the number of datasets on which the algorithm of this row produces significantly higher classification accuracy than the algorithm of this column. The denominator is the number of datasets on which the algorithm of this column produces significantly higher classification accuracy than the algorithm of this row.

The remarks made previously in Section 4.3 are reinforced again when 5 algorithms: CN2, ITI, C4.5, C4.5 rules and Nevprop are compared. It is impossible to find one outstanding algorithm which outperforms all others within this group. Interestingly, CN2's classification accuracy, averaged over the whole set of experimental data, is higher than those of C4.5 and C4.5 rules. The number of datasets on which CN2 produces statistically significant higher classification accuracy is much less than C4.5. All 5 algorithms produce significantly higher classification accuracy than OC1, LVQ and three of the TOOLDIAG classifiers. The result of the comparison between algorithms of Group 2 and Group 3 is presented in Table 15.

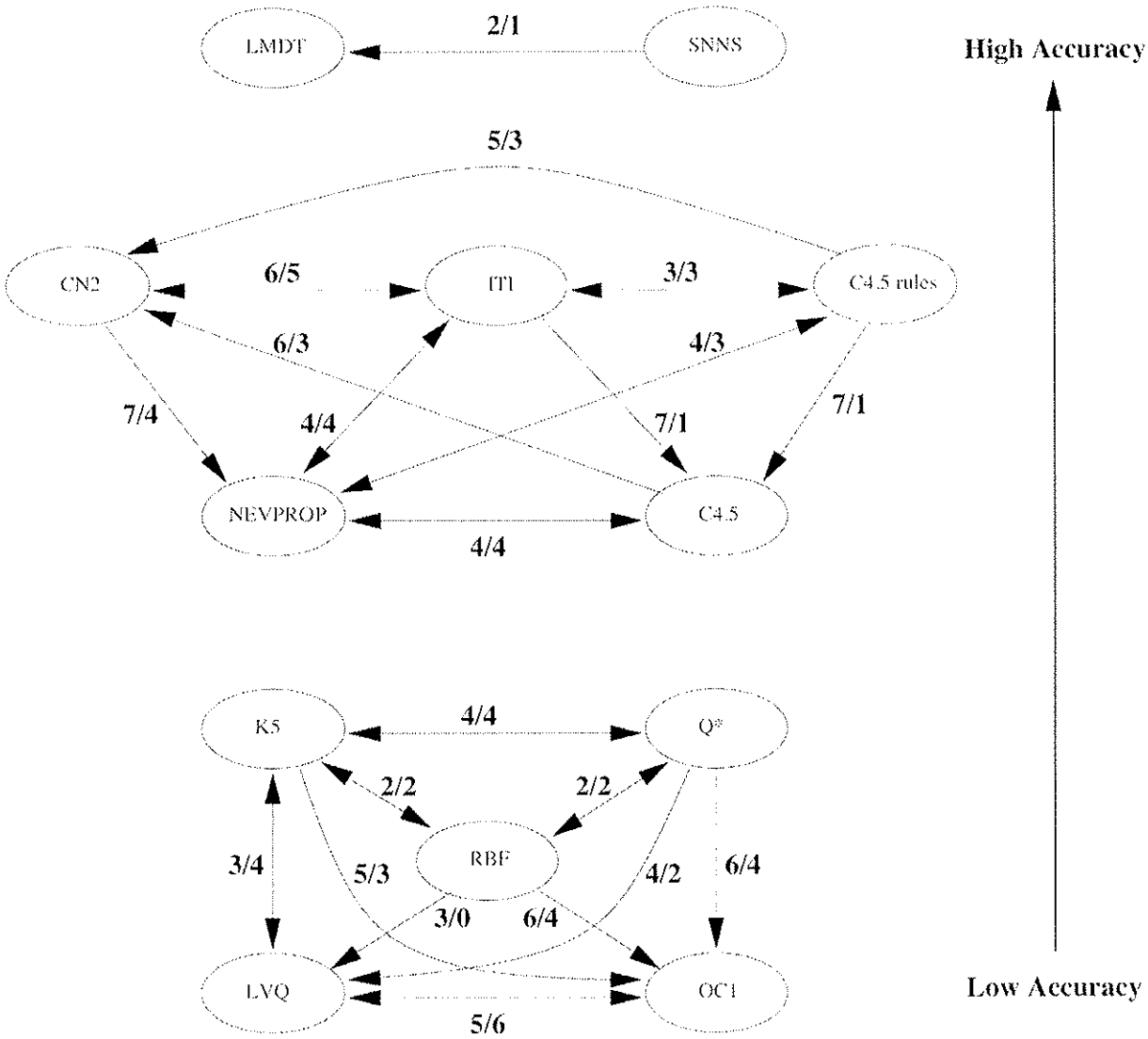

Figure 8: The classification performance of algorithms within each of 3 algorithm Groups 1, 2 and 3. The labelled ellipses represent each algorithm. Single-headed arrows between ellipses mean the first produces statistical significant higher classification accuracy on more datasets. Double-headed arrows represent 29approximately equal number of datasets on statistical significant higher classification accuracy are produced. Arrows are assigned fractions specifying how many datasets one algorithm produces with higher classification accuracy than another. In the case of single-headed arrows, the numerator belongs to the algorithm the arrow starts from and the denominator belongs to the algorithm the arrow points to.

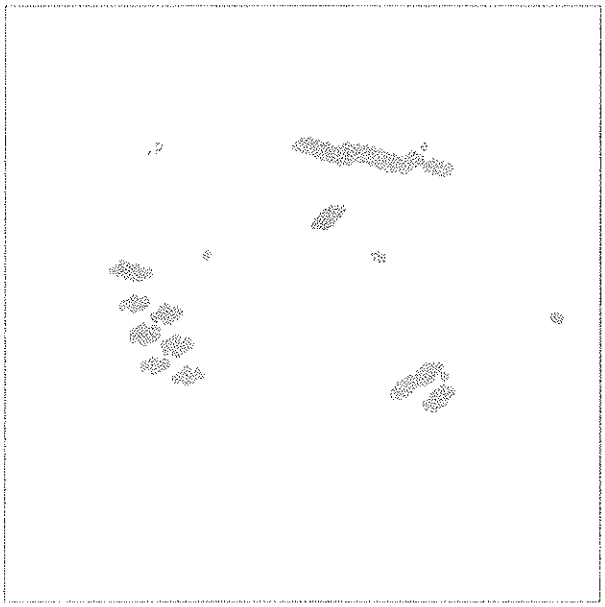

Figure 9: Mushroom data set (8124 cases, 22 attributes). Instances of each class are concentrated to form clear clusters in the projection. Maximal accuracy is 100%.

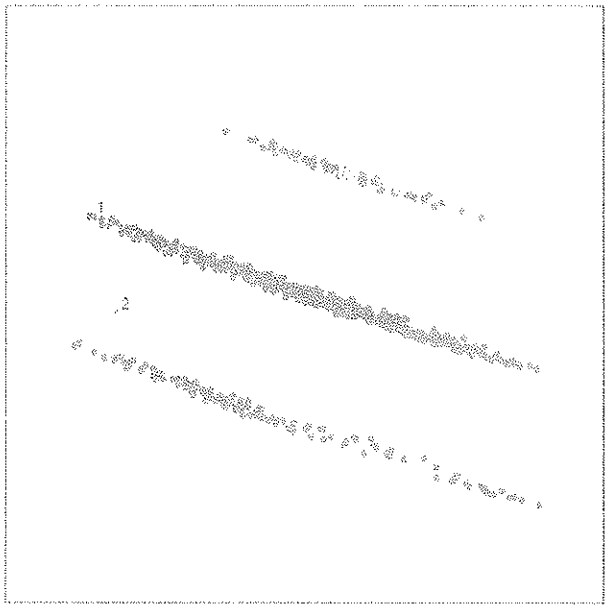

Figure 10: Tic-tac-toe data set (958 cases, 9 attributes, two classes). In the projection, classes form clusters in well separated regions with minor interleaving of two classes in one cluster (the top line). Maximal classification accuracy is 99.17%.

5 Data Analysis

5.1 Data Visualization

Visualization is a powerful technique in data analysis. It provides a number of facilities: structuring raw data, developing statistical models; and finding outliers. Visualization techniques can be used in several ways: explorative analysis, confirmative analysis, and presentation. Explorative analysis explores data without any pre-supposed hypothesis and usually performs undirected search for interesting structures and trends. Confirmative analysis starts from some hypothesis about the data and performs goal-oriented examination of the hypotheses. Presentation data visualization starts from *a priori* fixed facts to be presented and chooses an appropriate presentation technique to present facts in a high-quality visualization.

XGobi [27, 1, 4] is one of many visual techniques for exploring databases [21, 2]. It offers users a view of data in many forms: pairwise plots, three-dimensional rotation and grand tour (rotation in higher dimensions). In addition it supports directed search in higher dimensions for interesting views by the projection pursuit guided tour. Moreover, XGobi provides easily interpretable plots and multiple views of the same data which can be linked together. The approach is flawed in the sense that it relies on the existence of functional dependencies between attributes which may not exist. It also accommodates visualization data with missing values. XGobi is chosen to analyze our experimental datasets in order to confirm any hypothesis about the performance of the algorithms and explore interesting structures in the data. XGobi is a good candidate for our data exploration because, like the algorithms and data for our study, it is freely available on the Web[15] and satisfies the platform requirements for the experiment.

5.2 Results of Data Analysis

The result of data visualization on datasets where classification accuracy is very high (99%-100%) (achieved by LMDT, C4.5, C4.5 rules, IT1) shows that: the instances of each class are grouped to form distinct clusters of example instances by their class. These clusters also appear to be separately distributed in the attribute space (see Figures 9 and 10). In one or several regions, minor overlapping can be observed (in the Tic-tac-toe dataset for example). By contrast, in the Primary Tumor dataset, where the classification accuracy is poor (30%-50%), instances of 22 classes seem randomly distributed in 17-dimensional attribute space without clusters or clear regularities (see Figure 11).

[15] Two versions of XGobi can be obtained in:
`http://www.research.att.com/~andreas`

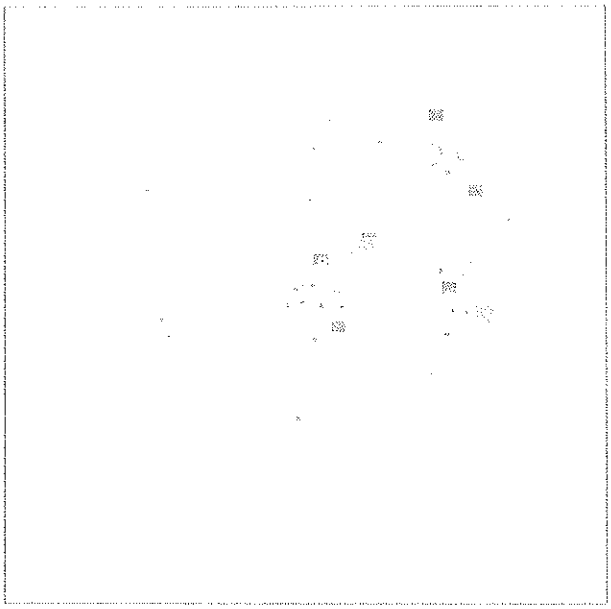

Figure 11: Primary tumor dataset (339 cases, 17 attributes, 22 classes). Each distinct symbol represents one class. In the projection, instances appear to be distributed randomly without clear regularity. Maximal accuracy is 49.2%.

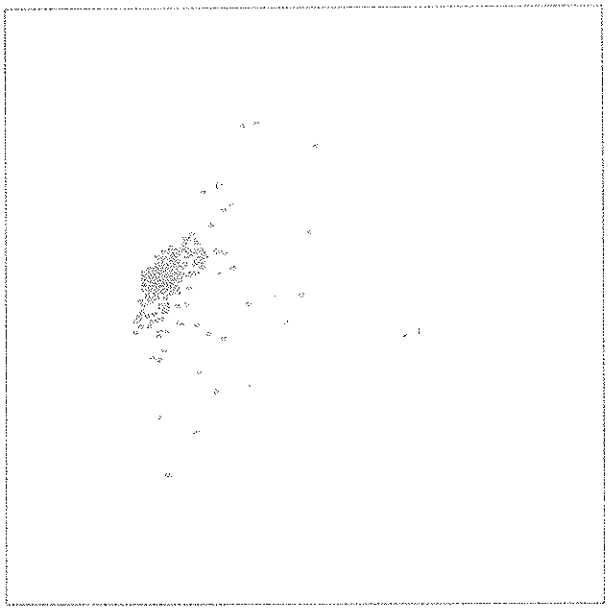

Figure 12: Breast cancer W. (Wisconsin) dataset (699 cases, 9 attributes). Two classes are distributed in two separated regions with minor interleaving in several places in the projection. Maximal measured classification accuracy is 95.74%.

In datasets where maximal classification accuracy is 90%-100%, e.g. Breast Cancer W., Iris, Image, Zoo and Wine datasets for example, instances of each class may (or may not) be clustered but are distributed into well separated regions with (or without) minor overlapping. Figures 12, 13, 14, 28, and 15 illustrate this phenomenon.

Further exploring datasets with decreasing maximal measured classification accuracy shows tendencies toward reduced clustering. In Lymphography and Cleveland datasets, where the maximal classification accuracies are 83%-85%, one can still recognize separate regions of class distributions in the projections. However, these class regions are often to be interleaved (see Figure 16). The situation deteriorates in the Bupa and Glass datasets, where maximal classification accuracy measured is 77.72% and 74.78% respectively. Although some regularities can be found in this dataset, in general classes are not clearly separated. In Bupa, the instances of two classes are mixed and concentrated mainly in one region. Consequently, the maximum classification accuracy on this dataset is not high.

Several special cases are worthy of attention. The first is the Lung Cancer dataset. Despite three classes distributed in several separated regions (see Figure 20), the maximal classification accuracy achieved (by K5) is low (68.54%).

Importantly, the Lung Cancer is the dataset with the smallest number of cases (32 cases) and the second largest number of attributes (56). This is why, although the distribution of classes in the projection is reasonable, the maximal classification accuracy is poor.

The second interesting dataset is the Balance Scale, where OC1 achieved a high classification accuracy. This dataset is the only one from an artificial domain and consequently has a clear but complex structure (see Figure 21). In this dataset, the contribution of each attribute in the class distribution seems evenly distributed. Further evidence is provided by the high performance of SNNS.

Another interesting result is observed in exploration of the impact of irrelevant attributes through visualization. Visualization is not a good technique for identifying irrelevant attributes but once identified their lack of contribution can be visually recognized. The first important observation is that in some datasets, several attributes make almost no contribution to class distribution. Figure 22 illustrates the impact of the different variables in the distribution of classes in the Glass dataset. While Na and RI attributes noticeably spread classes in these two dimensions, the contribution of Ka to class distribution is not significant. The situation is even worse for variable 2 in the Ionosphere dataset. While variable 6 alone almost perfectly spreads classes by its value (see Figure 23), a scatter plot of its combination with variable 2 shows variable 2 has only one zero value and therefore cannot be used for class recognition (see Figure 24).

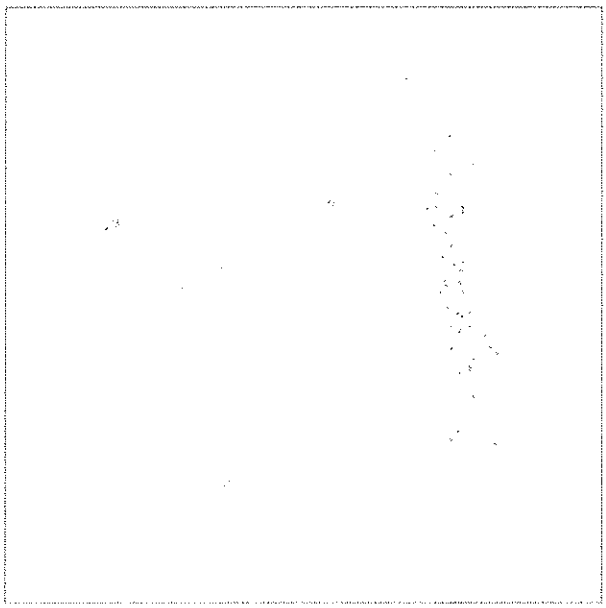

Figure 13: Iris dataset (150 cases, 4 attributes, 3 classes). Class 1 (to the right) is separated from two others in the projection. The remaining two classes are distributed in two distinct regions without obvious boundary between them. Maximal classification accuracy is 94.45%.

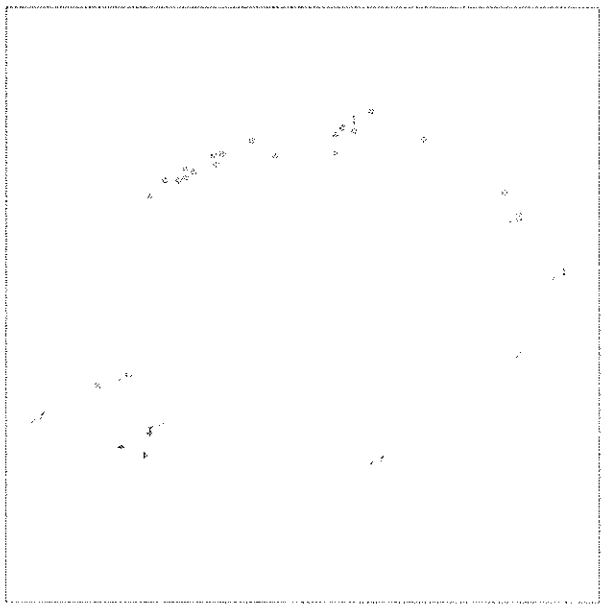

Figure 14: Zoo dataset (101 cases, 16 attributes classes). The majority of classes are distributed in well separated regions in the projection. Maximal measured classification accuracy is 96.61%.

Another important observation is that in some datasets, only several attributes separate classes. The Ionosphere dataset is an illustration. From 34 attributes, only 3 (variables 6, 5, and 3) separate classes (see Figure 25). The addition of the remaining attributes is unlikely to improve clustering and separation and in fact make a negative contribution (see Figure 26). The second similar illustrative example is the Wine dataset. From a total of 13 attributes, only 3 variables (7, 1, and 10) contribute to separate classes in reasonably well separated regions (see Figure 27). One more notable point is that both the above observations occur in datasets where the classification accuracy of OC1 and LVQ is significantly lower than that of C4.5, C4.5 rules, ITI and CN2. This signals a hypothesis on the impact of irrelevant attributes on OC1 and LVQ.

A final point concerns the impact of different decision tree pruning techniques on classification accuracy in combination with visualization. C4.5 uses an error based pruning technique and ITI a virtual pruning technique. While C4.5 prunes away subtrees (pruning which does not affect the classification accuracy of the resulting tree on hold-out subset), ITI only substitutes a pruned subtree by a leaf with a default class label and stores all cases in the form of a list of exceptions. In some cases, this technique can lead to differences in algorithm performance. According to the results of our statistical analysis, ITI achieves significantly higher classification accuracies than C4.5 on 7 datasets and C4.5 achieves significantly higher classification accuracy than ITI only on 1 dataset (Credit Screening). From the total 8 datasets 3 could not be viewed because they contain symbolic feature values. The 5 remaining are: Balance Scale, Hayes-Roth, Ionosphere, Pima Indian, and Tic-tac-toe. We found regions where instances of different classes mixed and there are uneven distributions of instances between classes in these regions. For example, the Balance Scale dataset contains only 49 instances of class 2, distributed irregularly among large numbers of instances of two other classes (each having 288 instances), distributed regularly in the attribute space (see Figure 21). In the Ionosphere dataset, instances of class 2 are likely to be tightly grouped to form 1 cluster, dominating the distribution of several instances of class 1 in this region (see Figures 26 and 25). In these cases, C4.5 will prune away subtrees corresponding to these minor instances while ITI preserves these instances in the exception list stored at the leaf node. Consequently, the resulting classification accuracies produced by ITI will be higher than C4.5. The situation is the same for the Tic-tac-toe and Hayes-Roth datasets. In Tic-tac-toe, some instances of class 2 are mixed with the majority of instances of class 1 into 2 small clusters (see Figure 29). In Hayes-Roth (see Figure 30), several instances of one class are distributed in a region, where the majority of instances are of other classes. In these cases, ITI will recognize exceptional cases while C4.5 wrongly classifies the instances as another class.

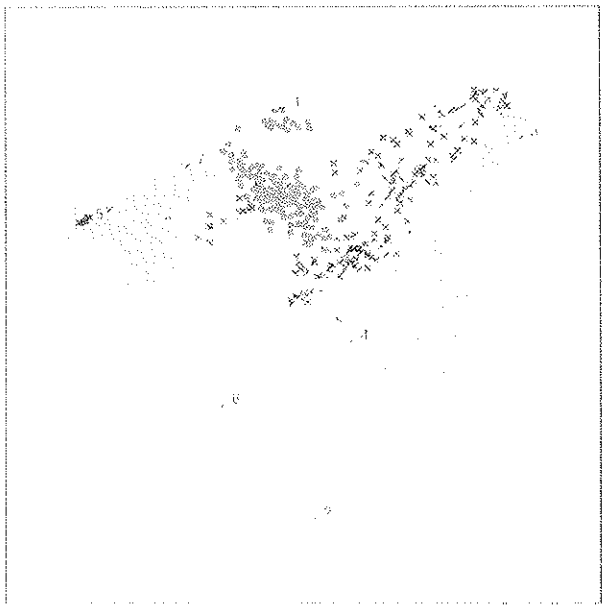

Figure 15: Image dataset (1497 cases, 19 attributes, 7 classes). Instances of the majority of classes are concentrated to form clusters (class 7, 2, 1, 4, 3) in the projection. Maximal classification accuracy measured is 97.61%.

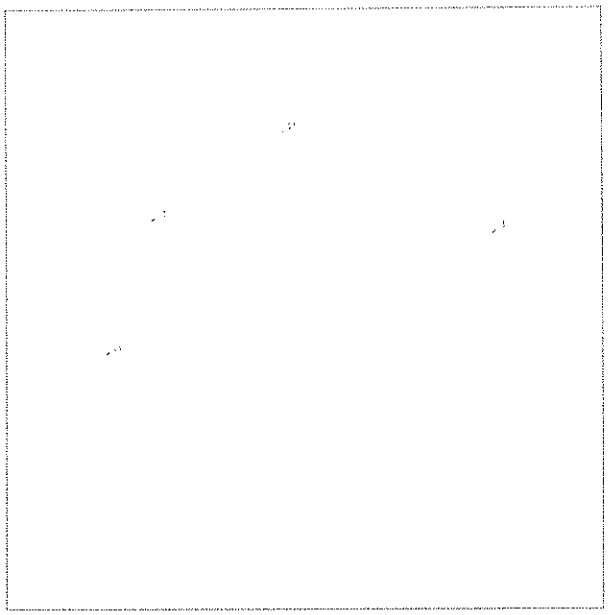

Figure 16: Lymphography dataset (148 cases, 18 attributes, 4 classes). Classes are allocated in several interleaved regions in the projection. Maximal classification accuracy measured is 84.34%.

5.3 Data Analysis using Feature Selection

Visualization is an interesting technique because data analysts can intuitively observe data distributions, find interesting structures, regularities and other useful information from raw data. However, in order to find out the contributions of some attributes in the data distribution, or the presence/absence of irrelevant attributes, the results of the analysis should be based on computation, with comparison in one or many aspects to ensure correctness. Therefore, visualization techniques alone are not sufficient and require additional support from other techniques. In this subsection, TOOLDIAG's feature selection functionality is used to find the presence/absence of irrelevant attributes in our datasets and their impact on the performance differences of machine learning algorithms.

5.3.1 TOOLDIAG and Feature Selection

Feature selection is one of the strongest aspects of TOOLDIAG. This facility allows the user to determine and select (based on several selection criterion) only those relevant features to classification from the initial dimensions of the feature space. Only features with the highest discriminative power will be selected and used for the classification task.

Feature selection consists of three components: *search strategy*, *selection criterion* and *stopping rules*. There are several search strategies such as *Univar*, *M-Covar*, and *MinErr*. The Univar strategy assumes all features are independent and each distributed in a unimodal way with its own mean and standard deviation. This strategy ignores the dependency of features and analyzes one feature at a time. In general, it calculates selection criterion for each feature independently, orders all features based on this criterion and selects them starting from the highest until the desired number of features are reached. The M-Covar strategy takes into consideration dependency among features and assumes features are multivariate and normally distributed. In general, it considers the pool of already selected features (controls). The feature joined to the pool is a feature with the highest selection criterion. The MinErr strategy makes no assumptions about the classes and features that define them. It keeps a pool of already selected features, and the remaining features as possible candidates. It will join a feature to those selected giving the lowest estimated classification error [9].

In TOOLDIAG, search can be driven as *Sequential Forward*, *Sequential Backward*, *Branch and Bound* or *Exhaustive*. Sequential Forward tests all features from the pool and joins those features that give the best criterion (together with the selected ones). The selected feature is then removed from the pool of available features. The search starts from the empty feature set and assumes all features as available. In reverse, the Sequential Backward Search deletes features from those available, those that give the best selection criterion, until the desired number

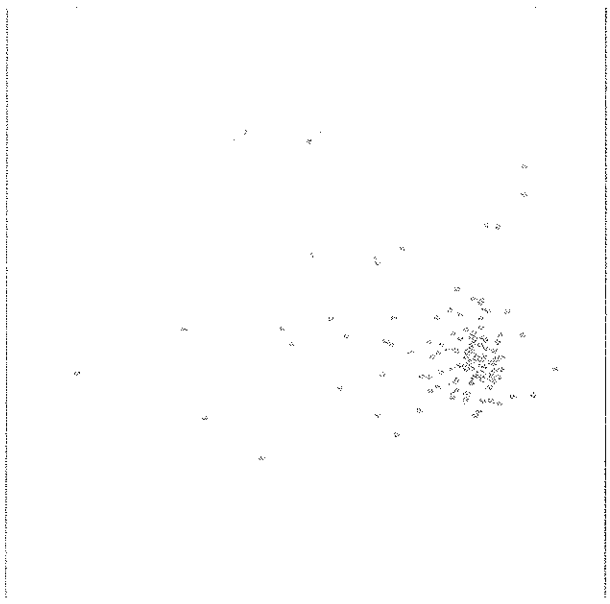

Figure 17: Bupa dataset (345 cases, 6 attributes, 2 classes). The majority of instances of both two classes are likely mixed in one region in the projection. Maximal classification accuracy measured is 77.72%.

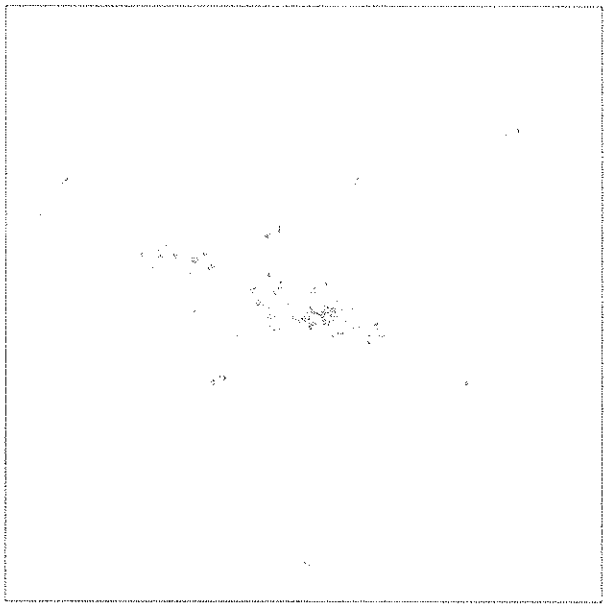

Figure 18: Glass dataset (214 cases, 9 attributes, 7 classes). In the projection, instances of some classes are grouped together but distributed in overlapping regions. Maximal classification accuracy is 74.78%.

of features is reached. Branch and Bound analyzes all possible subsets out of the entire feature set. The Exhaustive strategy searches all possible combinations. The search can be done for an incremental number of features permitting the user to find the global optimum of the selection criterion. The three selection criterion are: *minimum estimated error, interclass distance* and *probabilistic distance*. Minimum estimated error is a combination of feature selection and error estimation[16]. This criterion chooses those feature(s) that predict the smallest error for a certain classifier type[17]. Interclass distance criterion considers those features which maximize the distance among the samples of the classes (and minimize the distance within the sample class). Probabilistic distance chooses those features that separate the class conditional probability density functions of the classes, i.e. features that cause a minimal overlapping of the classes resulting in a good selection.

The next subsection gives the results of applying TOOLDIAG's feature selection facility on our study to identify the presence/absence of irrelevant attributes. The search strategy used is sequential forward. Selection criterion is minimum estimated error using 1-Nearest Neighbor classifier.

[16] Error estimation is a component of TOOLDIAG used to estimate an error rate produced by a classifier on a given dataset.
[17] The classifier is specified by the user from that available in TOOLDIAG (*k*-Nearest Neighbor, Radial Basis Function, Q*, etc.), familiar from earlier sections.

5.3.2 Irrelevant Attributes

The result of using TOOLDIAGs feature selection functionality to detect the presence of irrelevant attributes in the experimental datasets is tabulated in Table 16. Several explanations clarify the method used in calculating the number of irrelevant attributes. Firstly, in the experiment, we compute a number of irrelevant attributes, adding them to the pool of selected attributes does not increase and even reduces estimated classification accuracy, starting from the maximal values achieved so far.

For example in Bupa dataset, the result of feature selection is as follows:

```
---SELECTED FEATURES ---: 6
Nr. 1 = 3 ---Criterion = 0.46377
Nr. 2 = 5 ---Criterion = 0.64058
Nr. 3 = 4 ---Criterion = 0.64638
Nr. 4 = 6 ---Criterion = 0.67826
Nr. 5 = 2 ---Criterion = 0.63188
Nr. 6 = 1 ---Criterion = 0.62609
```

One can see that the estimated classification accuracy using 1-Nearest Neighbor starts from 0.46377 and is increased to 0.67826 when the number of selected features is increased from 1 to 4 (when features 3, 5, 4, and 6 are selected). Starting from this value, the addition of another 2 features (attributes 1 and 2) to the pool does not increase the estimated classification accuracy but reduces values to

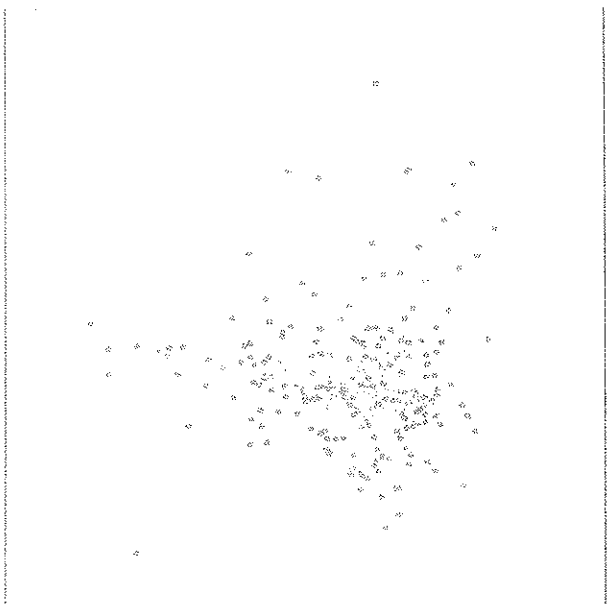

Figure 19: Pima Indian diabetes dataset (768 instances of two classes, 8 attributes). In the projection, instances of 2 classes likely overlap and concentrate in a single region. Maximal classification accuracy measured is 74.78%.

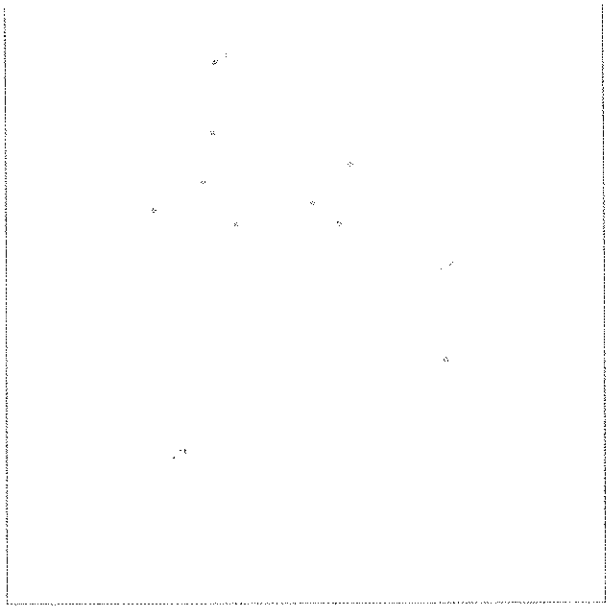

Figure 20: Lung Cancer dataset (32 cases, 56 attributes and 2 classes). Maximal classification accuracy measured is 68.54%.

0.63188 and 0.62609 respectively. In the case of several datasets, the addition of some features to the pool selected causes the estimated classification accuracy to fluctuate before a maximum value is reached. On these occasions, we count as irrelevant only those attributes selected after the maximal estimated classification accuracy was reached and conditionally consider all remaining attributes as relevant.

One observation is that the majority of datasets, where irrelevant features are found are those in which C4.5 and ITI achieve significantly superior classification accuracy than OC1 (C4.5 gives significantly higher classification accuracy than OC1 on 8 sets[18], ITI gives significantly higher classification accuracy than OC1 on 10 sets[19]). The second observation is that Balance Scale, where irrelevant features are absent, is the only dataset where OC1 achieves superior classification accuracy (except for LMDT and SNNS). However, the search for a correlation between the percentage of irrelevant attributes and the difference in performance of C4.5, ITI and OC1 shows that no relationship can be found.

The examination of datasets using feature selection also supports the observation made from visualization. In the Ionosphere dataset, using only 3 attributes (attributes 6, 5, and 3), the estimated classification accuracy given by the 1-Nearest Neighbor classifier is 92% and the estimated value is reduced to 86% when all attributes are used. This is confirmed in Figures 25 and 26. A similar case can be observed in the Wine dataset. From a total of 13 attributes, only 3 (attributes 7, 10 and 1) achieve 93% estimated classification accuracy. The addition of another 3 attributes (8, 11 and 3) makes no improvement to classification accuracy and only the addition of another 2 attributes (6 and 9) improves classification accuracy to its maximal value of 95%. From this point, the addition of remaining attributes causes estimated classification accuracy to decline.

5.4 Concluding Remarks

The results of data analysis combining visualization and feature selection leads to several noteworthy comments:

- the distribution of data in an Euclidean attribute space is an important factor in classification performance, contributing to the maximal classification accuracy achieved by machine learning algorithms. Obviously, in datasets where instances of each of the classes are grouped to form clean clusters in Euclidean space, the maximal classification accuracies achieved by learning algorithms are high. On the other hand, in datasets where instances of classes are distributed randomly in the Euclidean attribute space, maximal classification accuracies are poor. Mixed results are usually observed in datasets where classes are not well separated or distributed in overlapping regions.

[18] Glass, Image, Ionosphere, Lymphography, Pima Indian, Primary Tumor, Tic-tac-toe, and Zoo.

[19] Cleveland, Glass, Hayes-Roth, Image, Ionosphere, Lymphography, Pima Indian, Primary Tumor, Tic-tac-toe, and Zoo.

#	Database	Features	Irrelevant	Percentage
1	Balance Scale	4	0	0
2	Bupa	6	2	33.33
3	Cleveland	13	8	61.53
4	Glass	9	2	22.22
5	Hayes-Roth	4	1	25.00
6	Image	19	4	21.05
7	Ionosphere	34	31	91.06
8	Iris	4	1	25.00
9	Lung Cancer	56	36	64.29
10	Lymphography	18	3	16.67
11	Pima Indian	8	1	12.50
12	Tic-Tac-Toe	9	2	22.22
13	Wine	13	5	38.46
14	Zoo	17	5	29.41

Table 16: Irrelevant attributes detected by TOOLDIAG in the experimental datasets. Note: (i) Features: number of attributes. (ii) Irrelevant: number of irrelevant attributes. (iii) Percentage: percentage of irrelevant attributes.

#	Database	Irrelevant	No progress	Max. estimated	Estimated
1	Balance Scale	0	0	0.81	0.81
2	Bupa	2	0	0.67	0.62
3	Cleveland	7	0	0.78	0.58
4	Glass	2	0	0.74	0.73
5	Hayes-Roth	1	0	0.77	0.72
6	Image	4	5	0.97	0.96
7	Ionosphere	31	0	0.92	0.86
8	Iris	1	0	0.96	0.96
9	Lung Cancer	36	15	0.84	0.27
10	Lymphography	3	4	0.83	0.77
11	Pima Indian	1	2	0.69	0.67
12	Tic-Tac-Toe	2	0	0.81	0.69
13	Wine	3	3	0.95	0.76
14	Zoo	5	0	0.98	0.97

Table 17: The impact of irrelevant features on estimated classification accuracy.

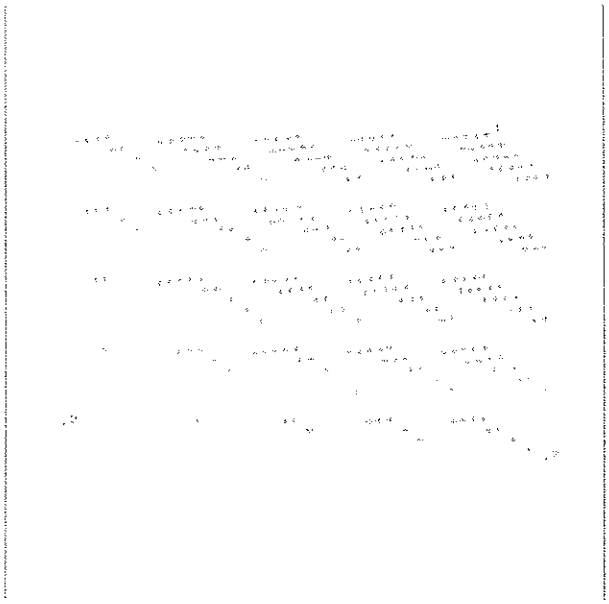

Figure 21: Balance Scale dataset (625 cases, 4 attributes, 3 classes). Data was artificially generated to model psychological experimental results. In the projection, data is complex but well structured with clear regularities. Maximal classification accuracy measured was 97.04%.

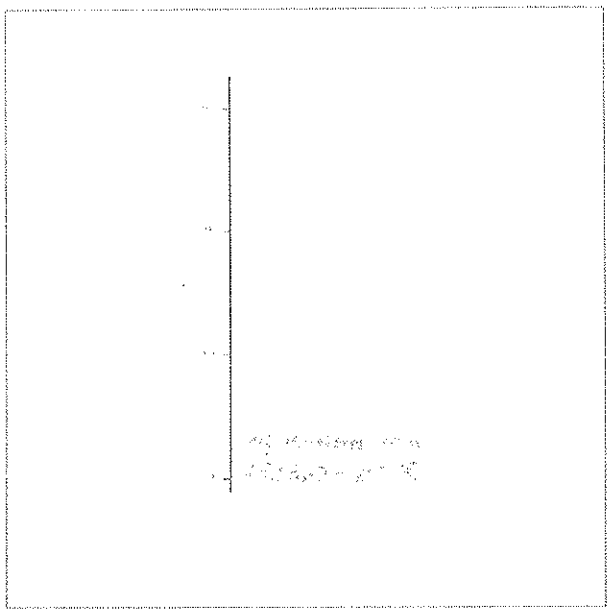

Figure 22: The minor impact of the variable "Ka" in class distribution in the Glass dataset. This attribute makes small contribution on the class distribution.

- in the Balance Scale dataset the distribution of classes has clear regularity but a complex structure. This is the only dataset which containing no irrelevant attributes. This is also the only dataset where OC1 and LVQ achieve significantly higher measured classification accuracy than C4.5, C4.5 rules, ITI, and CN2. Furthermore, the Lung Cancer dataset (where classes are distributed in well separated regions although the max-imal classification accuracies measured by algorithms is poor) is another dataset where OC1 and LVQ achieve significantly better classification accuracies[20]. Addit-ionally, in this dataset, the classification accuracies achieved by the TOOLDIAG classifiers are significantly higher.
- in several datasets the presence of irrelevant attributes can be observed visually. In some datasets, classes can be separated in distributed regions by using only a small number of attributes. In others, several attributes make no contribution to class distribution.
- the regions where instances of 2 classes are mixed, with a significantly uneven distribution of instances between two classes (the majority of instances are of one dominating class), are found in datasets where ITI achieves significantly higher classification accuracy than C4.5. This fact could be used as an explanation why virtual pruning (used in ITI) and error based pruning (used in C4.5) lead to differences in classifier performance.
- irrelevant attributes were detected using TOOLDIAG's feature selection functionality. In the majority of datasets with a high proportion of irrelevant attributes, C4.5, C4.5 rules, ITI, and CN2 achieved significantly higher classification accuracy than OC1 and LVQ.
- the impact of irrelevant attributes on classification accuracy, estimated using the 1-Nearest Neighbor classifier vary in the experimental datasets. No correlation between the percentage of irrelevant attributes and the differences in performance of the algorithms can be found.

[20] On this dataset, OC1 achieves superior classification accuracy to CN2 and ITI. LVQ gives significantly higher classification accuracy than C4.5, C4.5 rules, ITI, and CN2.

6 Experimental Results

The comparison of classification accuracy between 12 Machine Learning algorithms on 29 datasets from the UCI Repository leads to several noteworthy outcomes.

1. LMDT produces the highest average classification accuracies while OC1, LVQ, K5, Q* and RBF classifiers produce the lowest; this observation is independent of the data sources, the type of attribute values and the number of training instances;

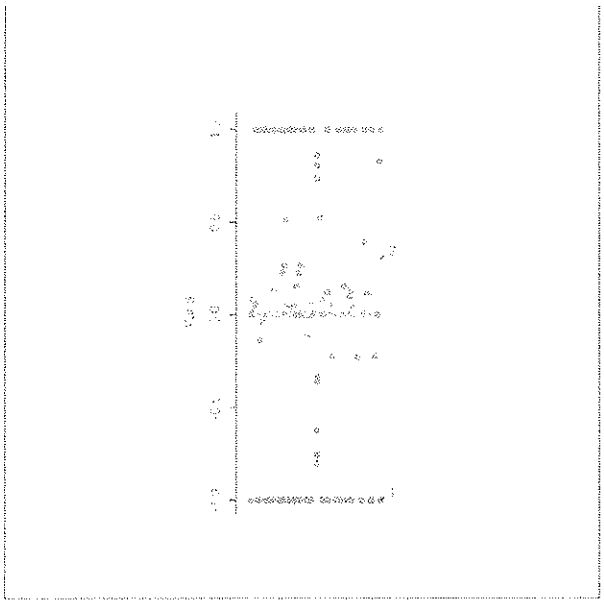

Figure 23: The significant impact of variable 6 in class distribution in Ionosphere dataset.

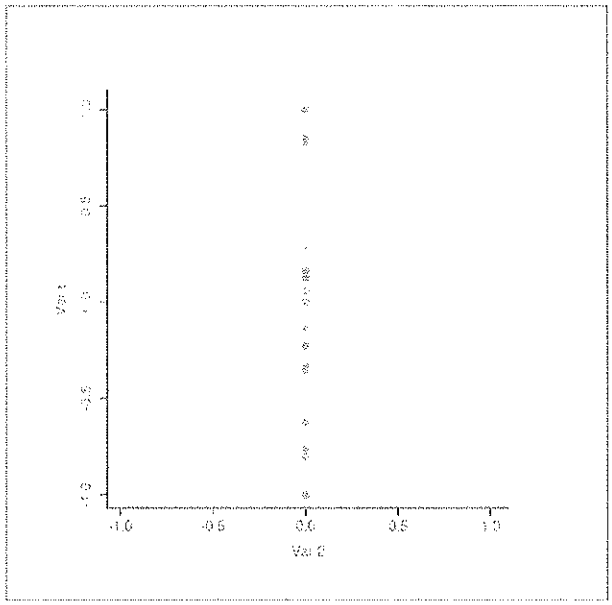

Figure 24: The impact of variable 2 in class distribution in Ionosphere dataset. This variable has a single value and therefore could not be used in class recognition.

2. the 12 algorithms selected for this study (C4.5, C4.5 rules, ITI, LMDT, CN2, LVQ, OC1, Nevprop, K5, Q* , RBF and SNNS) can be divided into three accuracy groups. Group 1 consists of 2 algorithms: LMDT and SNNS. Group 2 contains 5 algorithms: C4.5, C4.5 rules, ITI, Nevprop, and CN2. Group 3 consists of: OC1, LVQ, and K5, Q* and RBF.

3. within any single group, no algorithm with statistically significant higher classification accuracy can be ident-ified.

4. algorithms of Group 1 outperform all algorithms of Group 2 and 3. Algorithms of Group 2, in turn, outperform all algorithms in Group 3.

The observations are conclusive at first glance but warrant discussion. Although it is natural to assume that classification accuracies depend on data characteristics can we answer why and how this occurs? Secondly, what are the reasons LMDT outperforms other experimental algorithms? Why do OC1, LVQ and the TOOLDIAG classifiers produce the worse classification accuracy results? One possible explanation of the dependency of the classification accuracy of an algorithm on the dataset is the representative ability of the training data and distribution, and clustering of the underlying population. This is, in our view, one of the reasons why the performance of a classifier depends strongly on the characteristics of the dataset, especially training data used to build the classifier. If all classes of the underlying population are well separated and the training data perfectly represents this underlying distribution, classification accuracies for an algorithm are high. In contrast, if the underlying classes are randomly distributed or overlapping one can say categorically that no learning algorithm will achieve high classification accuracy on such data. These assumptions are supported by the results of data analysis using XGobi. In datasets where the maximal classification accuracy is very high, instances of classes are grouped. In datasets with lower measured classification accuracy, classes usually have no clear distinguishing boundaries or are distributed in overlapping regions in the attribute space. The worst classification accuracies are to be found in datasets where instances of classes are distributed randomly in the attribute space. Given a fixed dataset and a particular division into training and testing sets, why does one algorithm produce statistically significant higher classification accuracy than another? Firstly, data have underlying structure. If the classifier output by one algorithm better fits this underlying structure, the algorithms performance will be superior. In our experiment, the "Balance Scale" dataset is an example for this argument. In the Balance Scale the data distribution has a clear but complex structure. Moreover, class 2 has only 49 instances distributed widely in the attribute space, interleaved with instances of the 2 remaining classes (each has 288 instances, distributed regularly). When data can be characterised in this way, classifiers using multivariate testing (including neural networks) are more effective than those using univariate testing. A similar argument can be made for the Lung Cancer dataset which will be discussed later.

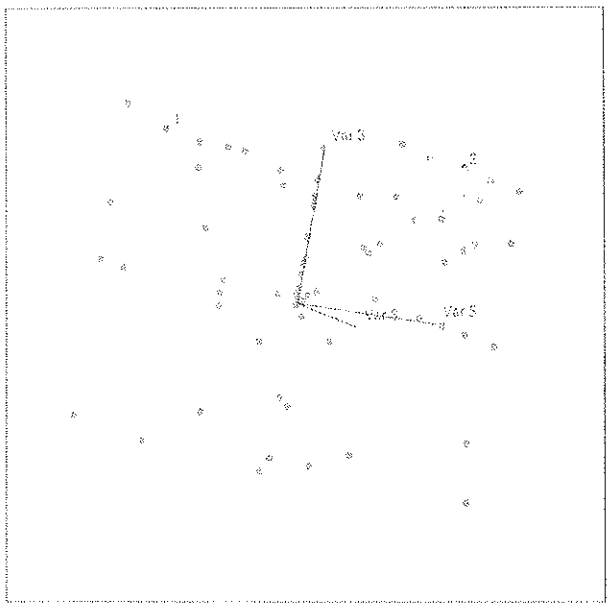

Figure 25: The impact of three attributes 3,5 and 6 to the class distribution in Ionosphere dataset. These three variables are likely to be good enough to distribute classes into separate regions.

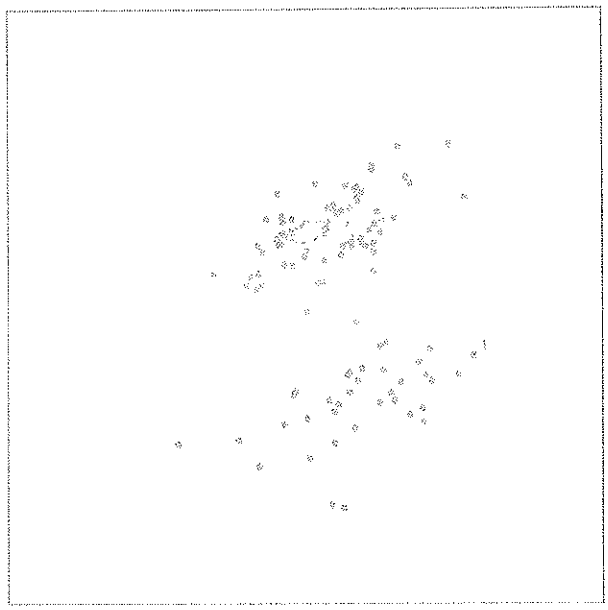

Figure 26: Ionosphere dataset (all attributes are used in the Projection Pursuit Guided Tour). Maximal classification accuracy measured is 93.65%.

Secondly, an important common characteristic of datasets used in this experiment is they all derive from natural domains and contain many irrelevant attributes. Irrelevant attributes always pose a significant problem for machine learning methods [26, 12, 24] and are another reason for performance discrimination. An algorithms capacity to deal with missing attributes has a profound impact on its classification accuracy. Several supporting arguments for this thesis can be found in the results of data analysis using visualization and feature selection.

Visualization also demonstrates that, in some datasets, several attributes make no contribution to the class distribution. In others, classes are well separated using only a limited number of attributes. The addition of remaining attributes for visualization causes the projection to deteriorate. Feature selection, particularly in terms of eliminating irrelevant features from the classifier, has an important impact on classifier performance.

In the decision tree classifier approach there are several ways of installing a test at a decision node. This includes univariate tests (C4.5, ITI) or multivariate tests (OC1). Generally, the OC1 oblique decision trees are smaller and at least as accurate as axis-parallel decision trees because the algorithm considers both axis-parallel and oblique splits of the training set. It is a fact that a decision tree, even an axis-parallel one, can be confused by too many irrelevant attributes [26]. This results because oblique decision trees learn the coefficient of each attribute at a decision node and values chosen for each coefficient should reflect the relative importance of the corresponding attributes. Clearly, the search for good coefficient values will be efficient when there are fewer attributes and the search space is therefore smaller [26].

Axis-parallel trees are constructed in a top-down manner by repeated searching for the most relevant attribute from the attribute space. Therefore, they are noticeably less sensitive to irrelevant attributes or noise [20]. This is one of the reasons for the inferior performance of OC1 in comparison with ITI, C4.5 and CN2[21]. A reasonable supporting argument can be found in results of data analysis by feature selection. Datasets, where univariate decision trees (C4.5, ITI) achieve significantly higher classification accuracies than multivariate testing decision trees (OC1) are datasets where irrelevant attributes are found.

Irrelevant attributes not only affect OC1 but also seem to be one of the reasons for the poor performance of LVQ and TOOLDIAG. This is because these algorithms treat all attributes equally to build knowledge structure. Furthermore, LVQ also requires the user to specify many important parameters during the learning phase to condition classification outcome. This process is non-deterministic, time consuming and cannot be well performed on a serial measurement condition (as in this experiment). Sub-optimal parameter values are an

[21] CN2 is not a decision tree but it searches a selector (the basic test on an attribute) for a single attribute one at a time. We therefore conditionally consider it as univariate testing.

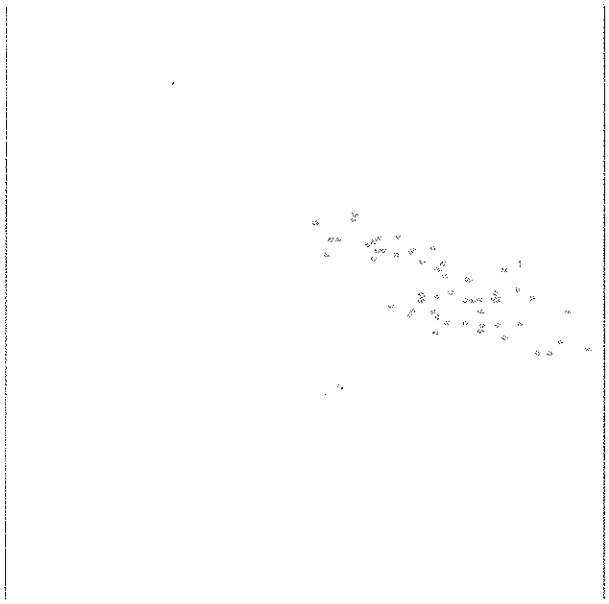

Figure 27: The impact of three attributes 7, 10 and 1 to the class distribution of the Wine dataset. These three variables appear sufficient to distribute classes in separated regions.

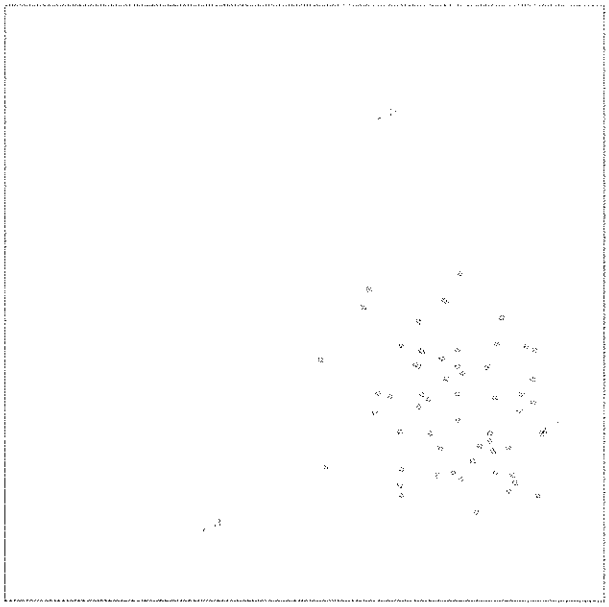

Figure 28: Wine dataset (178 cases, 13 attributes, 3 classes). In the projection, classes are allocated in three separated regions without clear boundaries. Maximal accuracy measured is 95.4%.

additional reason for the inferior classification performance of LVQ in our study.

It is interesting to discuss why LMDT outperforms all other algorithms. Firstly, at each node of the LMDT tree, a linear discrimination function is trained and installed. Discrimination functions are similar to multivariate tests installed at the node of an oblique decision tree and therefore achieve more accurate partitioning of the training instances. This line of reasoning leads us to predict that LMDT will suffer from the impact of irrelevant attributes. However, LMDT is equipped with a variable elimination mechanism to cope with such circumstances. Consequently, it is less affected by irrelevant attributes than other multivariate algorithms. In other words, LMDT performs well because it achieves the advantage of multivariate testing method at each decision nodes and, at the same time, inhibits the disadvantages of multivariate testing by using the variable elimination mechanism. From a statistical viewpoint LMDT represents the best machine learning algorithm (on balance) when the properties of the inter-dependence of attributes are unclear. This appears to be almost always the case in machine learning experiments where there is typically no time or expertise to commit to detailed data analysis.

It is also interesting to examine datasets where LMDT achieves significantly lower classification accuracy than other experimental algorithms. The number of these datasets is few. In the "Tic-tac-toe" dataset, C4.5 rules, CN2 and Nevprop outperform LMDT. In the "Ionosphere" dataset, LMDT is outperformed by 3 algorithms: C4.5, C4.5 rules and ITI. LMDT also gives worse performance than K5, Q* and RBF in the "Lung Cancer" dataset and worse than Q* and CN2 on the "Glass" dataset. One of the specific characteristics of Tic-tac-toe is that all its attributes are symbolic. In our experiment, these symbolic attributes are numerically encoded to make this data available to algorithms working on numeric data (OC1, LVQ, TOOLDIAG). The coding technique is rather simple: 3 symbolic attribute values x, o and b are substituted for three numbers 1, 2, and 3. LMDT treats these values as numeric in its learning phase and therefore its performance on this dataset is relatively poor.

In order to test this hypothesis, we compared measured classification accuracies of LMDT on original symbolic Tic-tac-toe dataset and numerically encoded data. The average classification accuracy over 5 runs on symbolic data was 97.6% and the standard deviation 0.44. The average classification accuracy over 5 runs on numerically encoded data is lower (81.6%) and standard deviation much higher (3.76). Presumably, this rough numerical encoding not only affects LMDT but also the performance of all other algorithms designed for numeric data including OC1, LVQ, and TOOLDIAG.

ITI, C4.5 rules and CN2 are less sensitive to this because discrete integers are treated in the same way as symbolic data. This phenomenon does not affect the neural network learning algorithms (Nevprop and SNNS) because integer values are binary encoded before being used for training.

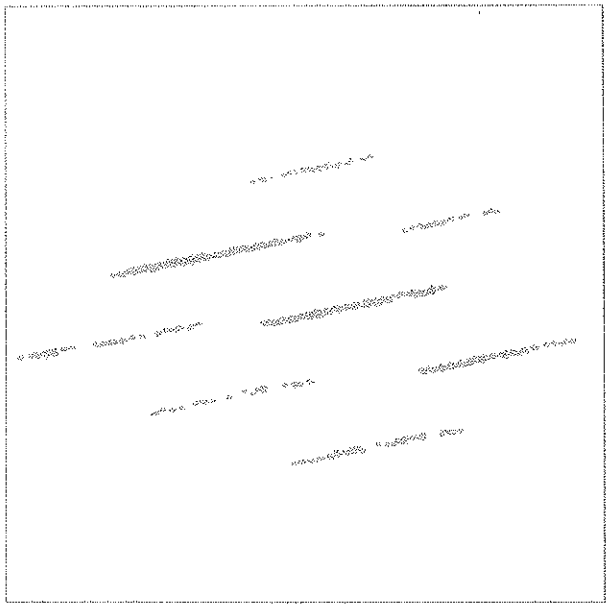

Figure 29: Another view of Tic-tac-toe dataset. Several instances of class 2 are mixed with large numbers of instances of class 1 in two clusters (top-left two clusters).

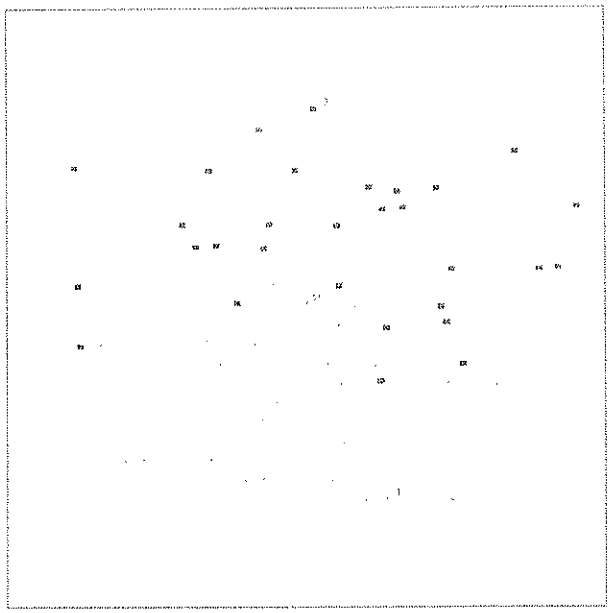

Figure 30: Hayes-Roth dataset (132 cases, 4 attributes, 3 classes). The figure shows loosely distributed classes. Maximal classification accuracy measured is 79.65%.

"Lung Cancer" is a dataset where all three TOOLDIAG classifiers outperform LMDT. Not only LMDT, these classifiers outperform all remaining algorithms. The remarkable characteristic of this dataset, mentiond previously, is that it is the dataset with the largest number of attributes and has the smallest number of instances. Furthermore, classes in this dataset are distributed in well-separated regions. Under these conditions, instance-based learning (TOOLDIAG classifiers as representative) are more effective than others because the data is insufficient to build a reliable knowledge structure. We might well conclude, if it were not for a single case as evidence, that instance-based learning is recommended when the number of training instances are few.

It is more difficult to argue why LMDT is outperformed by several algorithms on the "Ionosphere" and "Glass" datasets. Irrelevant attributes are present in these datasets and they contribute to the poor performance of algorithms using multivariate testing methods. However, LMDT seems less sensitive to such problems. Moreover, in these datasets, the impact of irrelevant attributes to classification accuracies estimated by TOOLDIAG 1-Nearest Neighbor classifier is not significant (see Table 17).

There are several notable inferences from these datasets. Firstly, the datasets contain only real-valued attributes. Secondly, in these datasets, Nevprop is the other algorithm producing significantly lower classification accuracy than almost all other algorithms[22]. One common behavior of these algorithms is that they normalize real-valued data in the learning process. This occurs automatically in LMDT and as a default parameter in Nevprop [24, 6]. Data normalization is valuable because it makes the scale of the feature values more or less the same for all features. This prevents the occlusion of important feature data if its values are small. However, normalization does sometimes destroy valuable information [18]. In addition, for some datasets, the bias of a univariate decision tree is more appropriate[23] and this is another reason why the multivariate testing decision tree produced by LMDT produces significantly lower classification accuracy than univariate testing decision tree algorithms (ITI and C4.5) on these datasets.

Over-fitting training data is a profound problem and avoiding it an important component of any learning

[22] Lower than C4.5, ITI, LMDT, CN2, LVQ, OC1, K5 and RBF on Glass. Lower than C4.5, ITI, LMDT, CN2, LVQ and OC1 on Ionosphere.

[23] According to Brodley (personal communication, 1997): although a univariate tree is a special case of a multivariate tree, LMDT's bias for finding such a tree may be inappropriate for some tasks because it may not find a univariate test when it should. LMDT's variable elimination method is a greedy search procedure and suffers from a problem inherent in any hill-climbing procedure: it can be deceived by local maxima. Therefore, although the hypothesis space LMDT searches includes univariate decision trees, the heuristic nature of LMDT's search may result in selecting a test from an inappropriate part of the hypothesis space [7].

algorithm. During training, the Q* algorithm adds a new prototype into the set of existing prototypes if the first prototype nearest the training instance wrongly classifies this instance. This addition results in an excessive number of prototypes which tune to the training data reducing the generality of the resulting classifier structure. This pathological behavior in Q* is the reason for high values of the classification accuracy on training data and low values on testing data.

Avoiding over-fitting the training data in decision trees is implemented using tree-pruning techniques. Each decision tree learning algorithm implements a particular pruning method to ensure the generality of the resulting knowledge structure. However, while all other decision tree classifiers prune away irrelevant subtrees, ITI uses a pruning in which a subtree is represented by a leaf with a default class and a list of exceptions. Each exception is an index to the list of instances and an indication of its non-default class label. In this way, the generalization of the model seems not to be improved but in some cases it may increase the classification accuracy produced by ITI's decision tree. This is one of the contributing reasons for higher accuracy than that achieved by C4.5 and OC1 decision trees. Our observations are supported through data analysis by visualization. The difference in classification accuracy between C4.5 rules and C4.5 is because C4.5 rules differs from C4.5 only in that the set of rules is produced from the unpruned tree. The generality of the knowledge structure in C4.5 rules is ensured by the use of a rule-post pruning technique [28].

7 Conclusion

This comparative study of 12 machine-learning algorithms on 29 datasets from natural domains was selected from UCI Repository of Machine Learning Databases. The algorithms are selected from 9 software packages namely: C4.5, ITI, CN2, OC1, LVQ, LMDT, SNNS, NEVPROP, and TOOLDIAG. The study covers a wide range of measurement and statistical analysis in order to identify the effectiveness of an algorithm on a dataset.

Our study reveals the outstanding performance of LMDT (and SNNS) on this data. This observation is independent of the originating source of the data, the type of attribute values and number of learning instances. We identify these 2 algorithms as Group

1. The SNNS backpropagation algorithm produces noticeably better classification accuracy but because of its specific characteristics[24] (it can run only on a limited number of datasets), its inclusion in Group 1 algorithms is conditional.

The remaining algorithms can be divided into two groups of relatively equal average classification accuracy. Group 2 consists of 5 algorithms: C4.5, C4.5 rules, ITI, Nevprop, and CN2. Group 3 consists of another five algorithms: OC1, LVQ, and three TOOLDIAG classifiers: k-Nearest Neighbour, Radial Basis Function Network and Q*. No algorithm within any one group is superior to another however each algorithm of Group 1 produces significantly better classification accuracy than algorithms of Group 2 and likewise Group 2 produces significantly better classification accuracy than algorithms in Group 3.

Naturally, the variance of algorithms wrt classification accuracy is due to the way an algorithm reacts to the data features. All experimental datasets are from natural domains (except one) and contain many irrelevant attributes and noise. Because LMDT combines both univariate testing methods for building the knowledge structure and a variable elimination mechanism for avoiding the effects of irrelevant attributes, it achieves the highest overall classification accuracy. The poor performance of OC1, LVQ, K5, Q* and RBF classifiers is due to the impact of irrelevant attributes, as well as over-fitting, and the difficulties of specifying optimal learning parameters.

[24] SNNS has no built-in data normalization mechanism and consequently requires external normalization process when working with datasets with numerical feature values, which are not on the same scale. Because this process should be applied on the whole dataset and could not be applied separately on training and testing sets, SNNS classification accuracy on these datasets could not be used in comparison.

References

[1] Swayne D. F. Hubbell N. Buja A. Xgobi meets S: Integrating software for data analysis. In *Computing Science and Statistics: Proceeding of the 23rd Symposium on the Interface*, pages 430–434, 1991.

[2] Cleveland W. S. Shyu M.J. Becker, R.A. The visual design and control of trellis display. *Journal of computational and Graphical Statistics*, 5:123–155, 1996.

[3] Gama J. Brazdil P. and Henery R. Characterizing the applicability of classification algorithms using meta level learning. In F. Bergadamo and L. de Raedth, editors, *Machine Learning ECML94*, pages 83–102, 1994.

[4] Buja A. Cook D. Manual controls for high-dimensional data projection. Technical report, Iowa State University and AT&T Laboratories, 1996.

[5] Wettschereck D. A hybrid nearest-neighbor and nearest-hyperrectangle algorithm. In F. Bergadamo and L. de Raedth, editors, *Machine Learning ECML-94*, pages 323–335, 1994.

[6] Department of Internal Medicine, Electrical Engineering, and Computer Science, University of Nevada, Reno. *Nevprop3 User Manual (Nevada backPropagation, Version 3)*, 1996.

[7] Brodley C. E. and Utgoff P.E. Multivariate versus univariate decision tree (technical report 92-8). Technical report, University of Massachusetts at Amherst, Amherst,

Massachusets 01003 USA, 1992.

[8] Utgoff P. E. Incremental induction of decision tree. *Machine Learning*, 4:161–186, November 1989.

[9] Rauber T. W. et al. A toolbox for analysis and visualisation of sensor data in supervision. In *93' International Conference on Fault Diagnosis*, Toulouse, France, 1993.

[10] Usama Fayyad et al. The KDD process for extracting useful knowledge from volumes of data. *Communication of the ACM*, 39:27–34, November 1996.

[11] Usama Fayyad and Ramasamy Uthurusamy. Data mining and knowledge discovery in databases. *Communication of the ACM*, 39:24– 25, November 1996.

[12] Helsinki University of Technology, Lab. of Comp. and Information Science, Finland. *LVQ_PAK The Learning Vector Quantization Program Package*. Version 3.1, 1995.

[13] `http://www.cs.uml.edu/shootout/`. Machine learning classifier shootout.

[14] Gama J. and Brazdil P. Characterization of classification algorithms. In *Proceeding of EPIA 95*.

[15] Michie D. Spiegelhalter D. J. and Taylor C. *Machine learning, Neural and Statistical Learning*. Ellis Horwood, 1994.

[16] Mingers J. An empirical comparison of pruning methods for decision tree induction. *Machine Learning*, 4(2):227–243, 1989.

[17] Murphy P. M. and D.W. Aha. *UCI repository of Machine Learning Databases*. University of California, Department of Information and Computer Science, Irvine CA, 1994.

[18] Duda R. O. and Hart P. E. *Pattern Classification and scene analysis*. John Wiley and Sons, New York, 1973.

[19] Clark P. and Boswell R. Rule induction with CN2 : Some recent improvements. In *Machine Learning - Proceeding of the Fifth European Conference (EWSL-91)*, pages 151–163, 1991.

[20] Clark P. and Niblett T. The CN2 induction algorithm. *Machine Learning*, 3(4):261–283, 1989.

[21] Keim D. A. Kriegel H. P. VisDB: Database exploration using multidimentional visualization. *Computer Graphic and Application*, pages 40– 49, 1994.

[22] Utgoff P.E. Decision tree induction based on efficient tree restructuring. Technical report.

[23] Utgoff P.E. A Kolmogorof-Smirnoff metric for decision tree induction. Technical report.

[24] Utgoff P.E. and Brodley C. E. Linear machine decision tree. Technical report.

[25] Quinlan J. R. *C4.5 : Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[26] Murthy S. K. Kasif S. Salzbeg S. A system for induction of oblique decision trees. *Journal of Artificial Intelligence Research 2*, 3(4):1–32, 1994.

[27] Buja A. Swayne D. F., Cook D. Xgobi: Interactive dynamic graphics in the X window system with a link to S. In *ASA Proc. of the Section on Statistical Graphics*, pages 1–8, 1991.

[28] Mitchell T. *Machine Learning*. McGraw-Hill, 1997.

[29] University of Stuttgart, Institute for parallel and distributed high performance systems (IPVR), Finland. *SNNS Stuttgart Neural Network Simulator. User manual. Version 4.1. Report No. 6/95*, 1995.