The Application of Machine Learning Techniques to Time-Series Data

A thesis
submitted in partial fulfillment
of the requirements for the degree
of
Master of Computing and Mathematical Sciences
at the
University of Waikato
by
Scott Mitchell

University of Waikato
1995

Abstract

"Knowledge discovery" is one of the most recent and fastest growing fields of research in computer science. It combines techniques from machine learning and database technology to find and extract meaningful knowledge from large, real world databases. Much real world data is temporal in nature, for example stock prices, dairy cow milk production figures or meteorological data. Most current knowledge discovery systems utilise similarity-based machine learning methods—"learning from examples"—which are not in general well suited to this type of data. Time-series analysis techniques are used extensively in signal processing and sequence identification applications such as speech recognition, but have not often been considered for knowledge discovery tasks.

This report documents new methods for discovering knowledge in real world time-series data. Two complementary approaches were investigated: 1) manipulation of the original dataset into a form that is usable by conventional similarity-based learners; and 2) using sequence identification techniques to learn the concepts embedded in the database. Experimental results obtained from applying both techniques to a large agricultural database are presented and analysed.

Contents

List of Figures

List of Tables

Acknowledgements

I wish to thank the following people for their assistance during the course of my research. Without their help, advice and support this project would not have been possible.

Lloyd Smith, my supervisor.
Robert Sherlock, for providing the database and correcting me on the finer points of dairy herd jargon.
Ian Witten, my other supervisor, for his helpful comments and suggestions.
Stephen Garner, Stuart Inglis and the rest of the WEKA team.
Andrew Donkin, Mike Vallabh and Dale Fletcher, for keeping the computer systems and printers alive under extreme pressure.
Doreen, for proofreading, provision of junk food, and just being you.
My parents, for all of their support and encouragement.

This work was partly funded by the New Zealand Foundation for Research in Science and Technology.

Copyright © 1995 by Scott Mitchell and the University of Waikato.

Chapter 1

Introduction

The amount of information stored in electronic databases around the world is enormous, and increasing at a rapid pace [4, 14, 37]. It has been estimated that the world supply of data is doubling every 20 months. Much of this data is collected automatically and is never seen by a human. The sheer quantity of data has caused many traditional analysis methods to become obsolete—they are too slow or inefficient to use on large databases, or knowledge of the problem domain is required that is not available from a large mass of automatically gathered data. Knowledge discovery (KD) provides new techniques for extracting meaningful knowledge from large, real world databases. KD is a relatively young field which has grown rapidly in recent years. It combines techniques from both machine learning (ML) and database technology to discover relevant features or relationships in data and present these in ways that are meaningful to human users [16, 5].

A significant amount of real world data is temporal in nature, in that the values of the variables are sampled at multiple points over some time period, so that the data stored for each entity has an additional "time" dimension. Examples of such time-series data are stock prices, dairy cow milk production figures or meteorological (wind and rainfall) measurements. Most existing knowledge discovery systems are built around similarity-based learning methods—"learning from examples"—which are not well suited to temporal data. Similarity based learners operate by generating descriptions of a concept

or classification over a set of examples, where each example describes a single object in the problem domain [21]. In a temporal database each object is described by a number of records—or alternatively, each of an objects' variables takes multiple values—along the time "axis". The mapping from time-series data to the examples used by the learning scheme is not obvious.

Time-series analysis techniques are well known in the signal processing world, where they play a part in applications such as speech recognition, data communications and image processing. In particular, sequence identification algorithms are used to detect and classify patterns occurring in a data stream. These methods include dynamic time warping (DTW) and hidden Markov models (HMMs). Sequence identification techniques have not been investigated in the context of knowledge discovery until very recently [4], but may have potential to be useful for tasks outside their traditional signal processing domain.

This report describes recent research into applying knowledge discovery techniques to time-series data. Experimental work was carried out using a large database of dairy cow milk production and performance measurements [36]. The primary aim of the research was to investigate and develop techniques for extracting knowledge from this database. Two different approaches were taken in this investigation:

1. Manipulation and restructuring of the original temporal data into a form that could be used effectively by conventional similarity based machine learning algorithms.

2. Adaptation of sequence identification methods to "learn" concepts described by a temporal database, and integration of these techniques into a similarity based learning system.

This research was undertaken as part of the WEKA Machine Learning Project [13, 22] at the University of Waikato. The objectives of the WEKA Project are:

1. To develop an interactive Machine Learning "Workbench" and the integration of new or novel ML schemes into this system.
2. To carry out case studies of the application of different machine learning techniques to problems in agriculture.
3. To extend the development and application of an entropy-based scheme for evaluation generalisations.

The research described here comprises part of Objectives 1 and 2.

The remainder of this report is structured as follows: Chapter 2 contains background information relating to knowledge discovery, machine learning, sequence identification and previous work combining these techniques. Chapter 3 describes the dairy cow database used for the experimental work, as well as the methodology for the experiments with both conventional machine learning schemes and sequence identification techniques. Chapter 4 contains a summary, analysis and discussion of the results for all experiments. Finally, Chapter 5 contains conclusions and a brief outline of proposed future work in this area. Complete, detailed results for all experiments may be found in Appendix A.

Chapter 2

Background and Review of Literature

While the goals and objectives of knowledge discovery and time-series analysis may have much in common, the two fields have not had much to do with each other until very recently. Knowledge discovery has developed largely from research in artificial intelligence and database technology, whereas sequence identification has generally been considered as a signal-processing application. There has been little crossover between the two areas so far. This chapter provides an introduction to machine learning, knowledge discovery and sequence identification techniques, and a review of relevant literature in these areas.

2.1 Machine Learning

Machine learning (ML) is the term used a encompass a wide variety of techniques used for the discovery of patterns and relationships in sets of data. The fundamental goal of any machine learning algorithm is to discover meaningful or non-trivial relationships in a set of "training" data and produce a generalisation of these relationships that can be used to interpret new, unseen data. Michie [24] defines the learning process as follows:

> *"A learning system uses sample data to generate an updated basis for improved classification of subsequent data from the same source, and expresses the new basis in intelligible symbolic form".*

This definition excludes a number of learning paradigms that do not produce symbolic descriptions, including connectionist methods, neural networks and genetic algorithms. For the purposes of the applications and methods described in this report only those techniques that do generate symbolic representations of knowledge. The output of a learning scheme is then some form of structural description of a dataset, acquired from examples of that data [21].

The knowledge learned by a machine learning scheme—the structural descriptions of the data—can be represented in different ways. Schemes such as genetic algorithms [11] or neural networks, mentioned above, generate implicit internal models of the data which are not easily understood by human beings or other machines. In this study we are concerned with being able to communicate the acquired knowledge to people, making the use of such schemes impractical. Other schemes generate more useful descriptions which can be interpreted by human users. These include:

Rules These are a popular form of knowledge representation as they resemble the way human experts tend to describe their knowledge of a domain. Rule representations range from simple "if-then" production rules [6] to more complex systems such as "ripple-down" rules and exception based schemes [9].

Decision trees and graphs The most basic form of decision graph is a simple binary tree, which is exactly equivalent to a set of if-then rules [28, 6] . Tree representations are not considered to be as comprehensible to humans as rules. However, newer schemes such as "exception dags" (directed acyclic graphs) [9] are claimed to be more natural and understandable representations.

Concept hierarchies Here the data is classified into a tree of categories and sub-categories. The topmost level of the tree represents the broadest

classification of the examples, this is, the most general description. Lower levels of the tree refine the initial classification, with the examples at the leaves of the tree having the most specific description [18, 8, 23]. A concept hierarchy differs from a decision tree in that examples may be placed at internal nodes of the concept tree. These examples are representative of more general relationships in the data—anomalous cases tend to be pushed down towards the leaves. In a decision tree all of the examples are classified at the leaves.

Machine learning schemes can be further compared along a number of different dimensions [21]. These dimensions tend to overlap to some extent but do provide a useful basis for comparison.

Supervised vs. Unsupervised learning This is one of the most fundamental distinctions between learning methods. Supervised learning involves developing descriptions from a pre-classified set of training examples, where the classifications are assigned by an expert in the problem domain. The aim is to produce descriptions that will accurately classify unseen test examples. In unsupervised learning, no prior classification is provided, and it is up to the learning scheme itself to generate one based on its analysis of the training data. Unsupervised schemes are often referred to as "clustering" schemes and are often closely related to statistical clustering methods [7, 12].

Similarity-based vs. Knowledge-based Similarity-based schemes form generalisations based solely on the similarities and differences found between the training examples. Knowledge-based learning makes use of a "domain theory" or background knowledge of the problem domain, usually supplied by an expert. If the domain theory is complete and correct, we have "explanation-based learning," where the scheme attempts to learn new and more efficient ways of interpreting new examples by analysing how existing examples are explained by the theory. In the case of an incomplete background knowledge the learner tries to improve the domain theory by correcting mistakes or adding new parts to the theory.

Top-down vs. Bottom-up A top-down scheme begins by examining a set of training examples, looking for generalisations that best describe the differences between the cases. In essence the scheme searches the space of possible concept descriptions for one which best matches the data. Bottom-up or "case-based" schemes look at the individual examples and build up descriptive structures from them.

Exact vs. Noise-tolerant Some schemes are able to cope with noisy input, such as missing or incorrect data. Similarity-based methods often fall into this category, since that look at large numbers of cases simultaneously and can average out some of the effects of noise. Knowledge-based schemes which analyse a single example extensively are more prone to failure when faced with inconsistent data.

As mentioned above, these distinctions are not orthogonal and there is much interest in hybrid schemes which combine several techniques to produce better overall performance [38]. Most schemes operate in batch mode, but it is also possible to have interactive learners, where the user serves as a "teacher", explaining new examples to the machine when the data fails to fit its existing model of the domain. These types of schemes are well suited to incremental learning, where new cases are assimilated without needed to re-process all previously seen examples.

2.2 Knowledge Discovery

Knowledge discovery (KD) is concerned with the extraction of meaningful knowledge from collections of real-world data, and communicating the discovered knowledge to people in an understandable way [16]. Typical knowledge discovery systems combine techniques from machine learning, statistics, artificial intelligence and database technology. These systems move beyond conventional machine learning in that they seek to discover knowledge that is both meaningful and relevant to specific groups of target users in a particular appli-

cation domain. The knowledge discovery process has been variously referred to as “database mining” or “database exploration” [21, 16].

Real-world data brings with it a new set of problems not generally faced in the idealised world of a pure machine learning task. Data is often noisy—sometimes extremely so—or poorly collected. There may be redundant or irrelevant variables present or, worse, important data might not have been collected at all. Large amounts of data are gathered automatically and might never been seen by a human user. In these cases prior knowledge of the domain may be deficient or missing entirely. Finally, real databases are often enormous—potentially thousands of variables and tens of millions of records may be present.

Brachman & Anand [5] consider the process of knowledge discovery as a human-centered task, with interaction between the user and machine at each step. They broadly divide the process into two parts—a data analysis and knowledge acquisition stage, and an application to make use of the discovered knowledge. The analysis phase may use a variety of statistical, machine learning or other tools, in tight interaction with a human data analyst. The idea is to find segments of the data worthy of further analysis, develop initial hypotheses about the data and determine which parameters of the data are relevant or useful. Knowledge discovery tools can then be applied to extract structural descriptions from the important subsets of the data. The analysis and discovery process relies on the domain knowledge of the analyst to determine what is useful and what is not. The discovered knowledge is encoded into a problem-solving system, where it can be accessed by end-users to solve real-world problems in their domain. The end result appears very similar to a traditional expert system, except that the knowledge embedded in the application was obtained from a combination of human domain knowledge and analysis, and machine discovery technology.

Most practical knowledge discovery systems to date have made use of similarity-based learning techniques for two main reasons:

1. Similarity-based learners are among the oldest, best developed and most

successful machine learning paradigms. Since they work with groups of examples at a time, similarity-based methods are often more immune to noise and work better with "messy" real-world data.

2. The output of a similarity-based scheme is usually some form of rules or decision tree. These knowledge representations are understandable by people and are easier to encode into an expert system or other problem solving environment.

Similarity-based schemes are not, in general, well equipped to deal with time-series data. The typical form of input for such algorithms is a simple two dimensional table of attribute-value pairs for each entity in the database. If the temporally-dimensioned data for an entity is presented to the learner as a set of examples, these will be treated as a group of distinct entities. Since the learner builds descriptions that can be used to classify a single example, any knowledge "discovered" from this input can only express coincidental relationships found between examples.

An alternative approach might be to concatenate all of the data for each object into a single large example, tracking the changes in each variable over a time period. This at least allows the learner to see all of the relevant data as single unit. However, many similarity-based schemes are only able to use zeroth-order logic in the structural descriptions they produce—each variable is considered to be independent of the rest and is examined in isolation. The resulting descriptions can only express relations comparing attributes which constant values. This is clearly not useful on time-series data, where a relation such as "production this week is greater than production last week" is both likely to occur and probably very relevant. Fortunately some similarity-based schemes, for instance Quinlan's FOIL (First Order Inductive Learner) [29] deal in first order logic and are capable of discovering these relationships.

However, a scheme like FOIL can still only cope with simple inter-attribute relations, for example $A \neq B$ or $C < D$. The data may contain relationships which are more complex to describe, such as $(A - B) > C$. In order for a

similarity based learner to detect this sort of pattern, it is necessary to generate "derived" attributes—new variables computed from the raw data values mcqueen:agricultural. For the previous example, a new variable containing the difference between the existing attributes A and B for each case would be added to the database. It is in this situation that the role of the human data analyst becomes crucial. Using their knowledge of the problem domain, the human expert must determine a set of raw and derived attributes that will—hopefully—enable the learning scheme to discover the useful patterns hidden in the data.

2.3 Sequence Identification

The task of discovering knowledge in time-series data can be modelled as a sequence identification problem. Essentially we are attempting to find "interesting" sequences or patterns in the data—where an interesting sequence might be one which is repeated often or at regular intervals, or an unexpected or anomalous sequence which may indicate the occurrence of a special event. The complexity of the pattern matching task varies depending on the amount of background knowledge about the time-series that is available.

In the worst case, we do not know what patterns we are looking for, nor where any "interesting" events are in the data. In this case the discovery task becomes completely unsupervised. The learning algorithm must decide for itself what constitutes an interesting pattern. Note that being unsupervised does not mean that the learning cannot be interactive. Interaction with a human user is still possible, with the person and machine working together to decide which sequences are important enough to be examined further.

More often the occurrences of important events—such as a significant drop in stock prices, or the onset of a hurricane—will have been recorded when the data was collected. This additional information can be considered as a classification of the data set. There will be a class for each type of relevant event covering all instances of that event, and a probably much larger "default"

class covering the remaining cases when nothing interesting was happening. Classified data can be attacked using supervised learning techniques to find patterns in the stream of data surrounding each event. If a pattern can be found that relies only on the structure of the time-series prior to a class of event, that pattern can be used to predict further occurrences of the event in the future.

In the best of all possible worlds, we would be given both completely classified training data, and a set of template patterns for each class. The learning task then decays into a pattern matching problem for classifying future data. Each of the templates can be matched against new data as it is generated, with the best match used to determine the classification at that time. Ideally we should be able to take incompletely classified data—as in the two previous cases—and apply suitable learning tools and domain knowledge to develop a description of the time-series that will allow us to classify unseen data in this way.

Humans are generally very good at detecting such patterns in time-series visually, but as with most such applications programming a machine to do so has proven to be a lot more difficult [4]. A large part of the difficulty lies in finding ways to detect and match patterns inexactly—with some notion of "fuzziness" to account for the inevitable noise and statistical variation in real-world data. Researchers in a number of fields have looked at this problem. The most relevant work appears to be in the area of signal processing, and in particular speech recognition. Good speech recognition schemes can reliably match words from their vocabulary despite variations in timing and pronunciation [4]. Two of the most successful techniques used recently in speech applications are dynamic time warping (DTW) and hidden Markov models (HMMs). These are described in more detail below.

2.3.1 Dynamic Time Warping

Dynamic time warping [33, 35] is a template-matching recognition method based around a dynamic programming algorithm. As such, DTW is a pattern matching method only, meaning that the templates must be generated externally, either by hand or using a discovery algorithm.

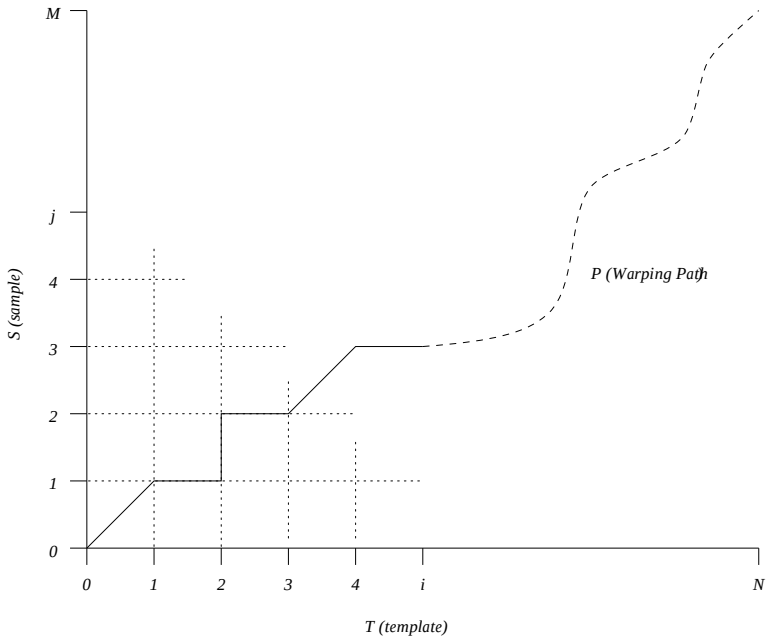

Figure 2.1: Example warping path

The basic aim of the dynamic time warping algorithm is to align the time axes of a sampled time series and a template, in order to minimise some distance measure. The time axis of either series is stretched or compressed to achieve the best possible fit, allowing a single template to match a range of "similar" patterns in the data. Both the template T and time series S consist of a sequence of data points over a time interval, which does not need to be the same for both sequences.

The algorithm matches each point s_i in S with one or more points t_j in T such that the overall distance measure between the two sequences is minimised. The process can be visualised as finding the least-cost path P through a grid of points, as shown in Figure 2.1. The grid axes correspond to the time axes of the two sequences, and each point (i, j) is assigned a weight which is just the distance measure $\delta(i, j)$ between s_i and t_j.

2.3.2 Hidden Markov Models

In contrast to the dynamic time warping approach, hidden Markov models are based around the idea of statistically modelling the signal or time series under

consideration [32]. The time series is characterised by a parametric statistical model which is configured so as to maximise the probability that the series could have been generated by that model. Since the model optimisation can be done algorithmically, a HMM system can adapt its own model to describe the data well. That is, it can learn from structure of the time series from a set of training data, and describe that structure by way of a statistical model. The model can then be used to classify or recognise new data. Thus, a HMM performs a vary similar task to a conventional supervised learning scheme, with the exception that it works on time-series data.

The theory of HMMs first appeared in a series of classic papers by Baum and his colleagues during the late 1960's and early 1970's [3] and was extended into speech processing applications in the 1970's by Baker at CMU and Jelinek and others at IBM [2]. The following overview of HMM theory and applications is of necessity relatively brief. For a fuller introduction to the subject the reader should consult any of [32, 19, 31, 34].

Markov Processes

A Markov process consists of a finite set of states and a set of possible transitions between those states. At regular, discrete time intervals the system undergoes a change of state, possibly back to the same state. A set of probabilities associated with each state determines the distribution of transitions from that state to each of the others. In the general case, a complete description of this system would require knowledge of all the states the system has been in up to the current time. In the special case of a first-order discrete-time Markov process, we assume that the current state of the model depends only on the previous state, that is

$$P[q_t = j | q_{t-1} = i, qt - 2 = k, \ldots] = P[q_t = j | qt - 1 = i] \tag{2.1}$$

where q_t is the state of the model at time t. This is known as the *Markov assumption*.

The process described above is called an observable Markov model, since the output of the system is the set of states over time, with each state corresponding

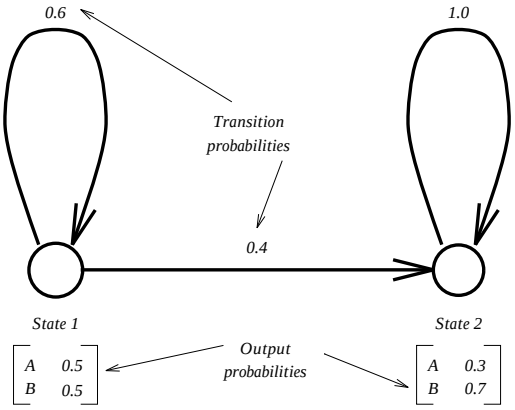

Figure 2.2: Simple two-state hidden Markov model

to an observable event. The output of the system is not random, making it unsuitable for many problems. A Hidden Markov model extends the system to include the case where the output produced is a probabilistic function of the state. Rabiner & Juang [32] describe the resulting system as

> *"a doubly embedded stochastic process with an underlying stochastic process that is* not *directly observable (it is hidden) but can be observed only through another set of stochastic processes that produce the sequence of observations."*

The "underlying stochastic process" is the state sequence which is no longer visible except indirectly through the output sequence. Figure 2.2 is an example of a simple two-state HMM with two output symbols and three possible transitions.

Elements of a hidden Markov model

A HMM using discrete output symbols can be completely specified by the following parameters:

N The number of states in the model, indexed with $1 \ldots N$. The state at time t is denoted by q_t.

M The number of output symbols, indexed as $1 \ldots M$. The set of the outputs themselves is written as $V = \{v_1, v_2, \ldots, v_M\}$.

A The state transition probability matrix. Each entry a_{ij} gives the probability that the model will make a transition to state j, given that it was already in state i, that is, $a_{ij} = P[q_{t+1} = j | q_t = i]$.

B The output probability matrix. Each entry $B_{ij}(k)$ gives the probability that output v_k will be produced when making a transition from state i to state j. If the output distribution is the same for all transitions out of state i the notation may be abbreviated to $b_i(k)$. The example HMM in Figure 2.2 associates different output distributions with the two transitions from state 1, so the full notation is required here.

π The initial state distribution, where each element π_i gives the probability that the model is in state i at time $t = 1$.

Since N and M are implicit in A and B, the complete parameter set of an HMM can be expressed as the vector

$$\lambda = (A, B, \pi) \tag{2.2}$$

Fundamental hidden Markov model problems

In general, there are three basic problems which must be solved in order for HMMs to be useful in real-world applications. These are:

The evaluation problem Given a model and a sequence of observations, what is the probability that the observations were generated by the model? The solution to this problem can be used to recognise unseen sequences, given a well trained model.

The decoding problem Given a model and a sequence of observations, what is the most likely state sequence for the model to have produced the observations? The solution to this problem is important in continuous speech recognition but probably less relevant in a supervised learning situation.

The learning problem Given a model and a set of observations, what is the best set of model parameters lambda that maximise the probability that the model produced the observations? The solution to this problem provides the means to "train" a model to recognise a particular sequence.

An overview of the solutions to all three fundamental HMM problems is presented below.

Problem 1—Evaluation

Here we are trying to calculate the probability of an observation sequence **O** given a model λ, $P(\mathbf{O}|\lambda)$. The obvious approach to this problem might be to enumerate all possible state sequences $\mathbf{q_1}, \ldots, \mathbf{q_n}$ of length T (the number of observations in **O** and calculate $P(\mathbf{O})$ for every such sequence. While this brute force solution will work it is very expensive in computational terms. For example, with $N = 5$ and $T = 100$ the complete computation requires around 10^{72} operations! Fortunately however there exists a simple inductive algorithm—the *forward procedure*—which solves the problem far more efficiently.

We define the *forward variable* $\alpha_t(i)$ where

$$\alpha_t(i) = P(o_1 o_2 \ldots o_t, q_t = i|\lambda), \tag{2.3}$$

the joint probability of the partial observation sequence up to time t and of the model being in state i at this time. The forward variable can be computed inductively as follows.

Initialisation:

$$\alpha_1(i) = \pi_i b_i(o_1), 1 \leq i \leq N \tag{2.4}$$

Recursion:

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(o_{t+1}, \quad \begin{matrix} 1 \leq t \leq T-1 \\ 1 \leq j \leq N \end{matrix} \tag{2.5}$$

Termination:

$$P(\mathbf{O}|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{2.6}$$

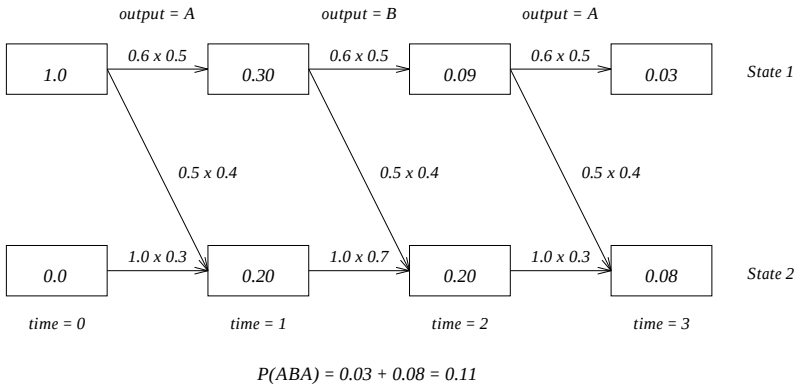

Figure 2.3: The forward procedure

The induction step sums the probabilities of reaching each state at time t_i from all possible previous states. The algorithm takes advantage of the Markov assumption, so that we only need to account for one previous state. The complexity of the forward algorithm is to the order of N^2T—for the example of $N = 5$ and $T = 100$ around 3000 operations are required, a saving of 69 orders of magnitude over the brute-force approach.

Figure 2.3 illustrates the calculation of α using the HMM from Figure 2.2 and the input sequence ABA. Each box show the cumulative probability of being in that state at that time, while arrows indicate legal transitions between states. The sum of the probabilities in the final column gives the overall probability of the observation sequence given the model.

Problem 2—Decoding

The forward algorithm gives the probability that an observation sequence was generated by a given HMM, but does not say anything about the underlying sequence of states. Knowledge of the state sequence is often useful, especially if the states have some correspondence with physical events.

Because the state sequence is hidden by definition, the best that can be done in practice to find the most probable state sequence to have generated the observed outputs. This can be achieved by a modification of the forward

algorithm, known as the *Viterbi algorithm*. Instead of just summing the probabilities of entering each state the Viterbi procedure remembers the most probable transition into the state at each time period.

We first define $\delta_t(i)$, the probability of the most likely path which 1) accounts for the first t observations and 2) ends in state i, as

$$\delta_t(i) = \max_{q_1 q_2 \cdots q_{t-1}} P(q_1 q_2 \ldots q_{t-1}, q_t = i, o_1 o_2 \ldots o_t | \lambda) \tag{2.7}$$

$$\delta_{t+1}(j) = (\max_i \delta_t(i) a_{ij}) b_j(o_{t+1}) \tag{2.8}$$

At each time step we need to remember the argument i that maximises $\delta_t(i)$ for each state. This information is stored in an $N \times T$ array m. The complete Viterbi algorithm proceeds as follows:

Initialisation:

$$\delta_1(i) = \pi_i b_i(o_1), 1 \leq i \leq N \tag{2.9}$$

$$m_1(i) = 0 \tag{2.10}$$

Recursion:

$$\delta_t(j) = \max_{1 \leq i \leq N} [\delta_{t-1}(i) a_{ij}] b_j(o_t), \quad \begin{matrix} 2 \leq t \leq T \\ 1 \leq j \leq N \end{matrix} \tag{2.11}$$

$$m_t(j) = \arg \max_{1 \leq i \leq N} [\delta_{t-1}(i) a_{ij}], \quad \begin{matrix} 2 \leq t \leq T \\ 1 \leq j \leq N \end{matrix} \tag{2.12}$$

Termination:

$$P^\star = \max_{1 \leq i \leq N} [\delta_T(i)] \tag{2.13}$$

$$q_T^\star = \arg \max_{1 \leq i \leq N} [\delta_T(i)] \tag{2.14}$$

State sequence backtracking:

$$q_t^\star = m_{t+1}(q_{t+1}^\star) \tag{2.15}$$

Problem 3—Learning

This is the most difficult of the three HMM problems. We wish to optimise the model parameter vector λ such that the probability of an observation sequence

O is maximised. While there is no known analytical method of achieving this goal, certain iterative procedures can be used to solve the problem. One such algorithm is the *forward-backward* algorithm, also known as the *Baum-Welch* or *expectation-maximisation* method.

We have already defined the forward variable $\alpha_t(i)$, the probability that the model is in state i at time t and has generated the first part of the observation sequence $o_1 o_2 \ldots o_t$. Similarly we can define the *backward variable* $\beta_t(i)$, the probability that the model is in state i at time t and *will* generate the remaining observations in the sequence, that is:

$$\beta_t(i) = P(o_{t+1} o_{t+2} \ldots o_T | q_t = i, \lambda) \tag{2.16}$$

As with the forward variable, the backward variable can be defined inductively:

Initialisation:

$$\beta_T(i) = 1, 1 \leq i \leq N \tag{2.17}$$

Recursion:

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(o_{t+1}) \beta_{t+1}(j), \quad \begin{matrix} t = T-1, T-2, \ldots, 1 \\ 1 \leq i \leq N \end{matrix} \tag{2.18}$$

We must also define $\gamma_t(i,j)$, the probability of taking the transition from state i to state j at time t:

$$\begin{aligned} \gamma_t(i,j) &= P(q_t = i, q_{t+1} = j, \mathbf{0} | \lambda) \\ &= \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{P(\mathbf{O}|\lambda)} \end{aligned} \tag{2.19}$$

Given this, we can say that the expected number of transitions from state i to j at any time is $\sum_{t=1}^{N} \gamma_t(i,j)$, and the expected number of transitions from state i to any other state at any time is $\sum_{t=1}^{T} \sum_k \gamma_t(i,k)$. From these results, the forward-backward procedure defines a set of re-estimation formulae for π, A and B:

$$\begin{aligned} \pi'_j &= \text{expected number of times in state } i \text{ at time 1} \\ &= \sum_k \gamma_t(1,k) \end{aligned} \tag{2.20}$$

$$
\begin{aligned}
a'_{ij} &= \frac{\text{expected number of transitions from state } i \text{ to } j}{\text{expected number of transitions from state } i} \\
&= \frac{\sum_{t=1}^{T} \gamma_t(i,j)}{\sum_{t=1}^{T} \sum_k \gamma_t(i,k)} \qquad (2.21) \\
b'_j(k) &= \frac{\text{expected number of times in state } j \text{ and observing symbol } v_k}{\text{expected number of times in state } j} \\
&= \frac{\sum_{t:y_t=k} \gamma_t(i,j)}{\sum_{t=1}^{T} \gamma_t(i,j)} \qquad (2.22)
\end{aligned}
$$

Equations 2.21 and 2.22 are both instances of the Baum-Welch algorithm [3]. Every such re-estimation of the model parameters is guaranteed to increase $P(\mathbf{O}|\lambda)$, unless a critical point—a local or global maximum—has been reached, in which case the estimate remains the same. The literature presents a number of proofs of the Baum-Welch method, including [3, 20].

Chapter 3

Experimental Methodology

A total of seven different experiments were performed on a database of dairy cow milking records supplied by the Dairying Research Corporation. The database tracks the performance of a small research dairy herd over part of the 1993–94 milking season. The experiments were designed to discover, recognise and describe time-series patterns hidden within the milking data. Both similarity-based learning and sequence identification techniques were used in the experiments.

This chapter provides some background information on the history of the dairy cow database and an overview of its structure. The methodology of each of the seven experiments is described in detail. Several issues that were important in the design and implementation of the experiments are introduced and discussed, including attribute selection, representation of sequences and the problems of skewed class sizes and missing values.

3.1 The Dairy Cow Database

The database used for the experimental work described in this report was obtained from the Dairying Research Corporation (DRC). Researchers at the DRC have developed an advanced dairy cow milking system—the Ruakura Milk Harvester (RMH)—that is currently deployed on around 15 farms [36].

Custom microcontroller hardware in each milking position is used to gather basic production and health-related data about each cow, at every milking. The system collects a large quantity of data that is clearly temporal in nature, as the performance of individual cows is tracked across the whole milking season.

Over the past three years Dr. Robert Sherlock—the designer of the RMH control and measurement systems—has been developing software to interact with the machine to collect and analyse milking data in real time. The software is intended to alert the operator to significant events and perhaps modify machine operating conditions as appropriate. This "MilkMAID" (Milking Monitor, Analysis and Information Display) system is in operation on an evaluation basis on four farms, as well as with the DRC's own research herd. The current system maintains animal and machine performance databases to a high level of reliability and allows the data to be displayed graphically. The next stage of development is to implement "intelligent" analyses of these databases to identify and report events of significance to the farmer as they occur [36]. Several examples of such events are described below:

Detection of cows "in heat". Most dairy herd reproduction in New Zealand is now performed via artificial insemination (AI), under the supervision of the Livestock Improvement Corporation. It is essential that the farmer knows when one of his or her cows is in heat so that the insemination can be carried out when the cow is at her most fertile. Generally the farmer makes this decision based on the behaviour of the animal in question, however a reliable automatic prediction could dramatically improve the "hit-rate" of insemination attempts.

Detection of mastitis. Mastitis is an infectious cattle disease that is relatively widespread amongst New Zealand dairy herds. The disease is easily treated in its early stages but is often hard to detect until it has developed into a more serious condition.

Detection of machine faults. Certain common failures in the milking machine are not easy to detect by simple inspection of the equipment, and may be hard to locate once they are known to occur. Any technique that could

identify possible fault locations with a good probability of success would obviously be welcomed.

The DRC was interested in investigating new methods of analysing the milking machine data. At the same time, the WEKA project was looking for new real-world data to work on, in particular data sets that could be tackled by sequence identification techniques. It was decided to concentrate on the first of the three DRC objectives, namely determining when a cow was in heat, for two reasons:

1. This was considered to be probably the simplest of the problems. A certain amount of domain knowledge already existed that could be used as a starting point for the investigation.
2. The data believed to be most relevant for the other two problems is not collected very reliably by the current milking machine design. In addition, the database already contains a classification for the first problem, indicating when cows were known to be in heat.

The DRC milking machine is currently in service on several farms, and is used with the DRC's own research herd. Most of this research has been carried out using the DRC herd data for the 1993–94 milking season. This herd contains 140 cows with records of two milkings per day over a period of 137 days between August and December 1993.

Although the herd milking database was supplied as a single file, it can be viewed as relational database structure with seven tables, as shown in Figure 3.1. The main components of the database are:

- A header record that identifies the farm, herd and season, as well as indicating the number of cows and milkings in the database.
- A table of summary information for each milking. Fields in this table include the milking date and time, the number of cows that took part, and the total volume of milk received.

- A table of cow definitions, one for each cow in the herd. The data for a cow includes its age, breed and the date of its last calving.

- The herd is divided into as many as 16 possibly overlapping subgroups, plus a "whole herd" group, for statistical purposes. There is a table containing summary statistics for each group over every milking. This table contains data such as the number of group members that took part in a given milking, and the mean and standard deviation of the milk volume for that group.

- The largest table in the database holds the individual cow milking records. These records track five primary variables:

  1. Milk volume
  2. Milking duration
  3. Milk conductivity
  4. Milk flow rate
  5. Order in which the cow entered the milking shed

  These quantities are recorded as percentage deviations from the herd mean for that milking. Running means and standard deviations of these percentage differences are also recorded. The milking records contain a number of binary variables, including the indication of whether the cow was in heat at a given time. Milking records are stored for every cow in the herd over every milking, regardless of whether the cow was actually present at a milking or not.

- A table associating a name with each herd subgroup.

- A table indicating which groups each cow is a member of. As mentioned above, a cow can be in more than one group and indeed every cow is implicitly a member of the 17th "whole herd" group.

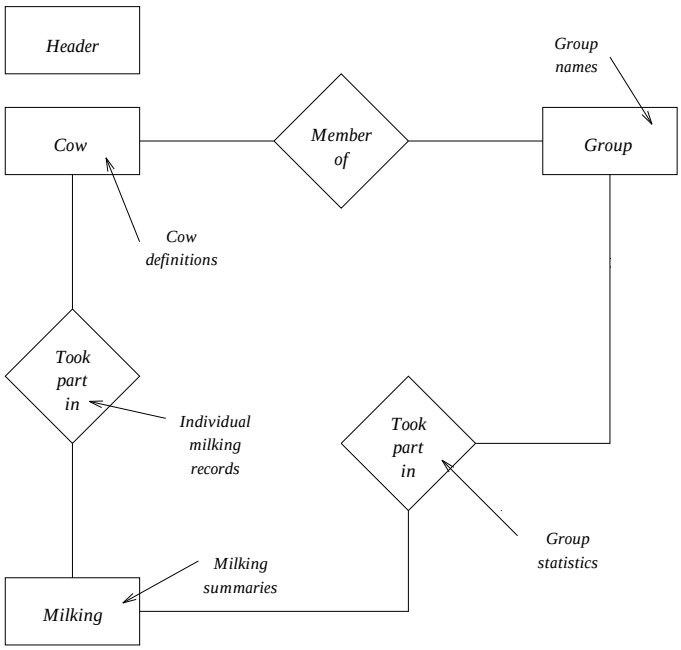

Figure 3.1: Organisation of the herd milking database

3.2 Database Conversion

The herd milking database was supplied by the DRC as a single file, in the format produced by the data gathering computer attached to the milking machine. The file was in a packed binary format that was very dependent on the architecture—an IBM compatible PC—and compiler—Borland / Turbo C—used by the data acquisition system. On the other hand, the WEKA workbench requires its input files to use the locally developed *ARFF* (Attribute-Relation File Format) representation [10]. ARFF is a human-readable text based format that is only able to represent a single flat table in each file. It was therefore necessary to perform some structural conversion on the database to render it into a form acceptable to the workbench schemes.

As a first step in converting the database, a program was written to convert the raw database file to a text based form. This program runs under MS-DOS on an IBM-PC compatible machine, with the output file being transferred to the UNIX-based Sun system used by the WEKA workbench. No information was changed by this process—the numbers are simply written out as text strings rather than the more compact binary form of the raw file. The advantages of a

textual representation are twofold: firstly the file is far more portable between machines and operating systems; and secondly a text file is much easier to edit and manipulate both by hand and with the various UNIX file processing utilities. The major drawback is that the file size is much increased—by a factor of approximately five in this case.

The DRC milking machine is still at an experimental stage in its development. Because of this the information in the database was not complete at the time the file was received. In particular, a large proportion of the "in heat" data from the individual milking records was missing. This data was entered by hand from a printed spreadsheet. Unfortunately the new data was given as a three-state enumeration—*yes*, *no* or *maybe*—whereas the database file represents this as a binary *yes* or *no* decision. After discussion with DRC researchers it was decided that the *maybe* class could be treated the same as the *yes* class if a binary classification was desired. However, the data stored in the textual database file still uses the three-state representation since this information may prove useful in later experiments.

An ARFF file consists of a header with the names and types of each of the variables, and a set of examples; thus it is essentially just a simple two-dimensional data table. Each of the experiments described below makes use of a different set of raw and computed attributes. In some cases subsets of the available examples have been used with a given attribute set. Programs and shell scripts were developed for each experiment to generate the required ARFF files from the plain text form of the database. These programs also perform the conversion of the "in heat" attribute from a probability to a binary value as described above. The operation of individual programs is described in more detail below where appropriate.

3.3 Similarity-based Learning Experiments

The first series of experiments with the herd milking database made use of similarity-based machine learning schemes. The aim of these experiments was

to determine if such schemes could be used to recognise and describe patterns in temporal data. The development of the experiments was more evolutionary than predetermined. The design of new experiments was influenced by the results of previous trials as more of the structure of the data became apparent. Complete results for all of the experiments described here can be found in Appendix A.

3.3.1 Attribute Selection

Probably the most important decision to be made in any experiment of this nature is to decide exactly which attributes should be included in the file given to a machine learning scheme. The attribute selection includes variables present in the raw database and "derived" attributes generated from existing variables. In the case of a relational database such as the herd milking data, producing input for a learning scheme will almost always involve some kind of projection or "flattening" of the database [22]. It is important that as little information as possible is lost by this process.

Initially, the only information available to guide the attribute selection process was the domain knowledge of the DRC researchers. Based on the available domain knowledge, it was decided to use only the attributes from the individual cow milking table, for the following reasons:

Firstly, the attributes considered important by the DRC for determining if the cow is in heat are in this table. These attributes are the volume percentage deviation from the herd mean, and the place in line that the cow entered the milking shed. Secondly, the group statistics table may contain interesting relationships common to the members of a group. However, since a cow may be a member of more than one group, flattening this table could generate data sets with huge numbers of attributes. Such data sets are impractical to deal with and often take an extremely long time to process with certain learning schemes. Worse, records with varying numbers of attributes might be generated when flattening the group table. None of the available learning schemes can deal with this situation.

Thirdly, the cow definition table is static—it contains no temporal data, so including any of these attributes would introduce a lot of redundancy into the data set. Redundant data is generally more difficult for similarity based schemes to deal with [21] so should be avoided. Finally, the milking summary table might be useful to detect the occurrence of global events affecting the whole herd, but using it would introduce redundancies similar to the cow definition data.

While no attributes from the other tables were used directly in any of the experiments, there is not reason why they cannot be used in the computation of derived attributes if desired.

3.3.2 The Schemes

The similarity-based experiments used two different learning schemes, *C4.5* and *FOIL*. These are described briefly below:

C4.5

C4.5 [30] is a system for inducing rules and decision trees from a set of examples. Much of C4.5 is derived from Quinlan's earlier induction system, ID3 [28]. The basic ID3 algorithm has been tested and modified by numerous researchers since its invention [25, 39]. However, C4.5 adds several new and interesting features that are worth mentioning:

- C4.5 uses a new *gain ratio* criterion for determining how to split the examples at each node of a decision tree. This removes ID3's strong bias towards tests with many outcomes. C4.5 also allows splits to be made on the values of numeric as well as enumerated variables.
- Decision trees induced by ID3 are often very complex and have a tendency to "over fit" the data. As a partial solution to this problem C4.5 introduces *pruned* decision trees. Pruned trees are derived from the original tree and

in general will lead to structures that cover the training set less thoroughly but perform better on unseen cases.

- C4.5 can also represent its learned knowledge in the form of production rules. Similarly to pruned decision trees, C4.5's rules are derived from the unpruned tree and are roughly equivalent to pruned trees in terms of classification accuracy.

FOIL

FOIL (First Order Inductive Learner) is another scheme developed by J. R. Quinlan at the University of Sydney [29]. It builds on concepts found in ID3 and Michalski's AQ to generate descriptions of logical relations using a subset of first-order logic. FOIL analyses a set of positive and negative examples of some relation and produces a structural description of the relation expressed as a set of Horn clauses. For data containing more than two classes, each class in turn is used as the source of positive examples, with the remaining data assumed to be negative examples. In this way a logical description of each class can be constructed. FOIL is able to express relationships between attributes, a feature that I hoped would be useful when analysing time-sequence examples.

The WEKA Prolog Evaluator

One of the most important features of the WEKA Workbench is that it allows many different schemes to be run on the same set of data and for the output of each scheme to be evaluated in a consistent fashion. The rules or decision tree produced by a learning scheme are translated into an equivalent Prolog representation and evaluated with respect to the training and test data sets. For each rule or decision tree leaf the evaluator indicates how often the rule or leaf is used, and how many examples it classifies correctly and incorrectly. The evaluator also provides a summary showing how many examples were classified correctly, classified incorrectly, classified into multiple classes or not classified at all. Many schemes—including FOIL—have only minimal internal

evaluation methods so the Prolog evaluator is immediately useful for analysing the output from these schemes. Additionally it is a valuable tool for evaluating the performance of different schemes on "neutral ground".

3.3.3 The Experiments

Experiment 1

The first experiment with the herd milking database was relatively simple, using no derived attributes and no re-arrangement of the data with respect to time. At this early stage I had no real knowledge of any structure in the data, how much noise was present or which attributes were the most relevant. The main purpose of the experiment was to make a rough "first pass" over the data and use the knowledge gained as a guide when developing later experiments.

The data set used for this experiment was the entire individual milking table from the herd database. All attributes were used with the exception of the unique cow and milking identifiers. The data set contained 38,360 examples with 27 attributes. The data was split into three classes based on the value the "heat observed" attribute.

Three different learning scheme "runs" were made using this data set. Firstly, in order to reduce the amount of data used for training, the data was split into 10 equal-sized, non-overlapping sections. For each of the sections, C4.5 was trained on the 10% subset and the descriptions it produced evaluated against the whole dataset.

For the second and third runs, C4.5 and FOIL were each trained and tested on the complete dataset. At this time the WEKA Prolog evaluator was not able to handle C4.5 rules so the results presented are for decision trees only.

Experiment 2

A number of changes were made to the data set for this experiment. Firstly the three-way *yes/no/maybe* classification from Experiment 1 was changed to a

binary *yes/no*, with the original *yes* and *maybe* classes merged together. The *maybe* class was very small—only 114 instances. From the farmer's point of view it is preferable to have an occasional "false positive" prediction than to miss times when a cow *is* in heat, so these examples were reclassified as *yes*.

Secondly, the two milkings for each cow on a given day were combined into a single example, thus reducing the number of instances in the data set to 19180. This was done mainly because the classification was made on a daily rather than individual milking basis, so it made sense to group the related milkings together. I hoped that this change would help to reveal relationships between the two milkings on a day when the cow was in heat.

Finally, several attributes were removed from the data set since they did not appear close to the top of the decision trees from Experiment 1. Attributes that are only used close to the leaves in a decision tree are *probably* only useful for making decisions specific to the training set and should not therefore be used to classify test cases. The retained attributes were the milking volume and the percentage deviations and running means / standard deviations for the volume, duration, milk rate, conductivity and milking order. These variables were included in each example for both milkings on that day. This gives a total of 33 attributes including the class.

Using this modified dataset, learning runs were carried out using C4.5 on 10% subsets of the data, plus C4.5 and FOIL runs on the whole file, as described in the previous experiment.

Experiment 3

In order to further investigate the idea of combining a time sequence of data into a single example, more modifications were made to the dataset. The milkings for each cow were split into a series of overlapping three-day periods, with the six milkings in each period making up an example in the dataset. The existing binary classification was retained, with an example classified as *yes* if the cow was in heat during the *middle* day of the time period. Clearly any descriptions

induced from this data can use future information and would not therefore be useful for prediction. However, I was trying to determine if the learning schemes could be used to detect a pattern across a time series, and felt that there were likely to be useful relationships in the future as well as past data.

With six milkings combined into one example the potential number of attributes was quite large. In order to have the learning schemes terminate within a reasonable amount of time more attributes were removed from each milking, again using the trees from the previous experiment to guide this decision. Eventually only the absolute volume and the five percentage deviation fields were retained for each milking.

The final dataset contained 6300 examples with 37 attributes including the class. The same learning runs were performed as for Experiments 1 and 2.

Experiment 4

The experiments described so far only use variables present in the raw herd milking database. These attributes simply record the physical measurements taken by the milking machine for each cow. The concepts hidden within these raw attributes are not necessarily in an "obvious" form that can be spotted by a similarity-based learner.

Specifically, the milking database records statistics for an individual cow relative to the performance of the entire herd at a particular time. Variations amongst the individual animals may be treated as random noise by the learner or worse, as very specific cases to be handled separately. Either strategy can lead to highly inaccurate descriptions by producing structures that far too general or too specific.

A possible solution to this problem is to make use of *derived attributes*—new variables computed or derived from the existing data. Derived variables can be as simple as the sum or difference of two attributes, or far more complex, for example a filtering or domain transform operation on a variable. The set

of possible derived attributes that *could* be generated is infinite. For a given application the attribute set must be chosen carefully to accentuate features of the data without obscuring or eliminating relevant information. It may also be appropriate to retain some or all of the original raw attributes alongside the derived variables. A combination of derived and raw attributes were used for the remainder of the similarity-based learning experiments, described below.

A problem that became particularly apparent during the previous experiments was the large disparity in size between the *yes* and *no* classes in the data. In the raw database, milkings when a cow was observed to be in heat account for only around 1.5% of the total data. This huge difference allows a learning scheme to produce a rule classifying every instance as *no* and still be correct 98.5% of the time. C4.5 is particularly guilty of this. It was decided to investigate this phenomenon to see if it could be controlled or at least discover how and why it was happening.

Experiment 4 extended the use of time-sequenced data in the learning examples. Instead of the three day segments of data used in Experiment 3, week-long (seven day) samples were used. The samples for each cow were generated by sliding a seven day wide "window" along the series of milkings for the cow and extracting the milkings within the window at each step. This procedure is shown in Figure 3.2. The contents of each window formed the basis of a single example in the resulting ARFF file. Examples were classified as *true* if the cow was in heat during the final day of the window, or *false* otherwise. As shown in the figure, successive windows overlap by six days, so it is possible for an "in heat" event to occur within a window that is classified as *false*.

The Experiment 4 dataset used the following raw and derived attributes for each of the 14 milkings in an example:

- The raw volume percentage deviation from the herd mean.
- The volume percentage deviation of this milking from the previous-but-one milking of this cow. This is the deviation from the previous morning for a morning milking, or the previous afternoon for an afternoon milking.

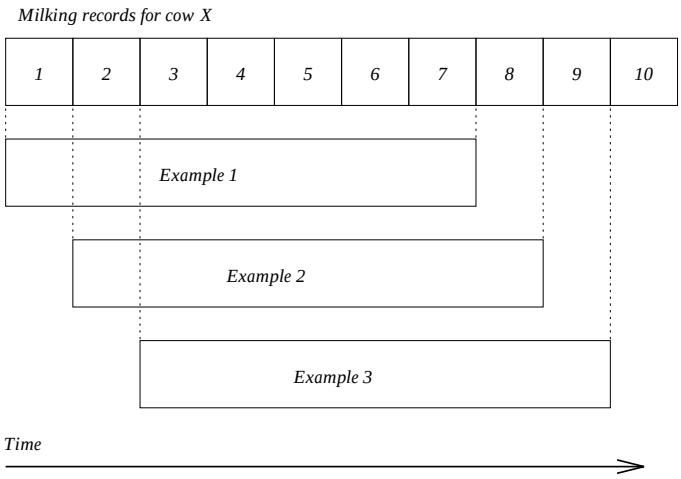

Figure 3.2: Sliding window for extracting time-series examples

- The volume percentage deviation from the mean of the morning and afternoon milkings in the window. The deviation from the morning mean was used for a morning milking, and the afternoon mean for an afternoon milking.

These attributes were chosen to model the performance of a cow with respect to itself as well as the whole herd. The complete dataset has 43 attributes including the class, and 14037 instances—368 *true* and 13669 *false*.

In addition to the complete dataset, a set of six smaller datasets with reduced numbers of negative examples were generated to investigate the effect of class size disparity on the results. The ARFF file generation program produced the reduced files by randomly selecting a specified percentage of the negative examples that it found. All of the positive examples were included as before. The new datasets contained respectively 1%, 2%, 5%, 10%, 20%, 50% and 100% of the negative examples (100% is the complete dataset). The ratio of positive to negative examples in the full dataset is just over 2.5%, so the 2% set has the closest to an even mix of classes.

C4.5 and FOIL were run on each of the seven datasets. In each case the scheme was trained on the full file and results evaluated against the training data by the WEKA Prolog evaluator. The output from the six reduced datasets was also evaluated against the full 100% set. The C4.5 evaluations were carried

out using each of the unpruned trees, pruned trees and rules produced by the program.

Experiment 5

The methodology used for this experiment is identical to that of Experiment 4—C4.5 and FOIL runs on seven datasets with varying proportions of negative examples. The differences in this experiment lie in the attribute selection and in the way the samples were generated.

The raw and derived attributes used in the Experiment 5 datasets were as follows:

- Raw milking volume.
- Raw volume percentage deviation from herd mean.
- Raw "milking order," This is the point when the cow entered the milking shed, expressed as a percentage of the whole herd.
- Raw running mean and running standard deviation of the volume and milking order deviations.
- The difference between the volume deviation and its running mean, expressed as the number of running standard deviations from the mean.
- The difference between the milking order and its running mean, expressed as the number of standard deviations from the mean.
- The ratio of the morning and afternoon milking volumes for each day.

These attributes are replicated for each of the 14 milkings in an example.

In the previous experiment the overlapping of sample windows meant that many negative examples also contained the "in heat" event that identifies a positive example. The Experiment 5 ARFF file generator was modified to prevent this situation from occurring. In the new generator program, after

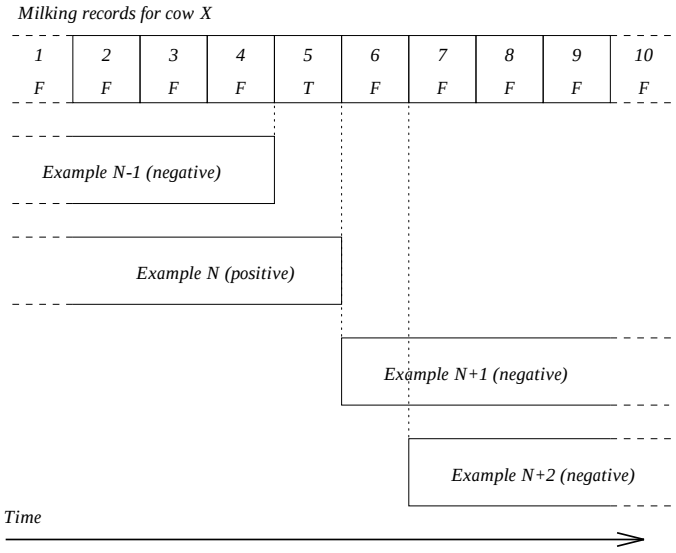

Figure 3.3: Modified sliding window for positive examples

a positive example has been generated the window is slid forward by the full seven days so that there is no overlap with the previous sample. The change is illustrated by Figure 3.3. This change was intended to increase the differentiation between patterns in the two classes—if the same pattern occurs in both positive and negative cases, the learner will have difficulty formulating a description that can distinguish between the two when the same pattern is seen in a test example.

The complete Experiment 5 dataset has 99 attributes including the class, and a total of 14548 examples—371 positive and 14177 negative.

I also made a second set of learning runs over the Experiment 5 datasets, with one additional derived attribute added. The new attribute indicates the number of days that have passed since the cow was last in heat, relative to the final day of the sample. This variable appears as a missing value for any examples before the first positive one for each cow. Using this attribute might appear at first to be perilously close to “cheating”. However, it turns out that this attribute covers a reasonably wide range of values across all the positive examples, so the best we can do is to say that if the value lies *outside* this range then it must be a negative example. More information is required to make the determination as to whether an example is positive.

The second set of Experiment 5 trials will be referred to as *Experiment 5a* for the remainder of this report.

3.4 Sequence Identification Experiments

This set of experiments used hidden Markov modelling techniques to learn the structure of the dairy herd database. Markov modelling has been used with considerable success for learning patterns in one-dimensional time-series such as speech signals. The objective of these experiments was to investigate whether or not these techniques could be used to form useful descriptions of the multi-dimensional data in the herd milking database.

3.4.1 The Hidden Markov Model Toolkit (HTK)

HTK is a comprehensive package of hidden Markov model research software developed by the Cambridge University Engineering Department Speech Group [41, 42]. The HTK system is intended primarily for speech recognition applications, and contains numerous features that are specific to this purpose. However, the heart of HTK is a reasonably general Markov modelling package that can in theory be used to model time-sequenced processes other than speech. The system is composed of approximately ten separate programs and a library of low-level support routines. Between them these programs provide the necessary functionality for

1. Preprocessing and encoding of time-series data using Linear Predictive Coding (LPC) techniques.
2. Initialising and re-estimating the HMM parameters from a set of training examples.
3. Testing the recognition performance of the trained model on a set of unseen test sequences.

The system allows individual models to be combined in arbitrary networks to represent complete phrases and sentences as well as individual words and sub-words, making it a very powerful and flexible package.

3.4.2 Attribute and Model selection

As with the previous experiments using similarity-based learning, the choice of which attributes and records from the database to use is very important. Unlike the similarity based schemes, HTK expects each "example" that it is given to consist of a time-sequenced set of values for one or more variables. The individual values in the sequence are known as *samples*, with the complete sequence referred to as a *sample file*. A sample file usually begins life as a raw digitised speech signal. This signal is generally encoded by some form of LPC algorithm into a multi-dimensional parameterised representation of the raw data. A network of HMMs can then be trained and tested on the parameterised sample data.

An extra complication added to the HMM experiments was the need to define both initial "prototype" models and the way these were combined into a network for the recognition program. It was decided to use a single model for each class in the data, that is, a model for *true* and a model for *false*. The recognition network was defined so that the recogniser was forced to make an absolute decision between *true* and *false* for each test sequence. Defining the system in this way gives experiments that are conceptually similar to those using similarity-based methods, despite the radically different learning algorithm being used.

HTK uses a prototype model to define the number of states and initial transition probabilities for the HMMs used in an experiment. The transition probabilities and output probability distributions are updated by the model initialisation and re-estimation programs to reflect the data in the training samples. The number of states is not changed however. It is possible to use different prototypes for each model, or for subsets of the models. In the interests of keep-

ing the experiments simple a single prototype was used for both models in each experiment. The designs of the prototypes are described in detail below.

3.4.3 Experiment 6

This first sequence identification experiment used the same set of attributes as Experiment 5a, the final similarity-based experiment. In Experiment 5a a series of milkings for an individual cow were combined into a single "wide" example for the learning scheme. This approach is not necessary with HTK, since each member of the time-series can become a separate element in a sample.

Another conversion program was written to convert the data used in Experiment 5a into sequences suitable for use with HTK. This program used the same group of raw and derived attributes from each milking, and extended the "length" of the sequences from five days to seven. Each milking became a separate sample in the sequence. Thus each sequence was made up of a sequence of 14 consecutive milking records for a given cow. Sequences were classified as *true* if an "in heat" event was recorded during the seventh day of the sequence, or *false* otherwise. As for Experiment 5a the sequences were arranged to be non-overlapping following the occurrence of a positive example. The final dataset contained 13996 sequences with the following attributes in each:

- The class—*true* or *false*.
- Raw milking volume.
- Volume ratio for the day of the milking.
- Number of days since the last "in heat" event for the cow.
- Volume percentage deviation, running mean and standard deviation, and number of standard deviations from the running mean.
- Milking order percentage, running mean and standard deviation, and number of standard deviations from the running mean.

Unlike the similarity-bases schemes used in my earlier experiments, HTK has no concept of a "missing value" in its data. Missing values occur when a milking is missing from the database. For this first experiment the missing data was simply replaced with zeroes—this is in fact what was done in the raw database file. Obviously this simple approach is not ideal since it adds noise to the sequences where the missing values occur. However, the number of missing milkings in the database is not large—around 5% of the total milkings—so the effects were not expected to be too serious. Possible alternative solutions to this problem are to eliminate sequences containing missing values, or to somehow interpolate the missing data from surrounding milkings. the first of these alternatives has been used in Experiment 7, described below.

The dataset for Experiment 6 contains 11 different values plus the class in each sample, so can be considered multi-dimensional data. The HTK LP coding tools only support single-dimensional data, thus it was not practical to LP encode the milking samples before passing them into the Markov modeller. Each sequence was simply treated as an 11-dimensional parametric representation of a time-series pattern. I did not consider this to be a serious problem since the herd milkings are *not* speech data and the Markov modelling system is capable of dealing with arbitrary sequences. It would be possible to LP encode each attribute separately, but this would result in an explosive increase in the number of attributes and the complexity of the models—this could not be justified during the initial stages of the investigation.

Four different prototype models were used, with differing numbers of states and initial transition matrices. The complete definitions for all of the prototype models used can be found in Appendix B. The most important parts of the prototype transition matrices are the elements that are set to zero. These elements will never be changed by the modelling system, and thus define the set of possible paths through the model. The exact choice of non-zero values in the remaining matrix elements is not crucial, as these values will be modified by the re-estimation process to best fit the training data.

The HTK system has a hard-coded limit on the number of samples it can

process at once, so it was impossible to use all of the available negative examples. A subset of the negative examples was randomly chosen, of approximately the same size as the set of positive examples. The final dataset was made up of 359 positive and 344 negative sequences.

The prototype models were trained on a randomly chosen two-thirds (66%) of the dataset. A preliminary model initialisation run was made to configure the output probability distributions. The complete parameter sets of each model were then updated using the Baum-Welch re-estimation algorithm. The parameter re-estimation step was performed 10 times for each model. After this many iterations, none of the model parameters were changed significantly by further re-estimation. The resulting models were tested on the remaining one-third (34%) of the data.

3.4.4 Experiment 7

This experiment used a methodology similar to that of Experiment 6, with some significant changes to the structure of the input sequences and prototype models.

The sequences in this experiment used data from a single day's milkings, instead of the relatively short sequences of multi-dimensional samples in the previous experiment. Raw and derived attributes from the two milkings of a cow on a given day were combined into a long sequence of 64 one-dimensional samples. All of the 25 available raw attributes from the milking records were used, along with the following derived attributes for each milking:

- The number of days since the cow was last in heat.
- The ratio of the AM and PM milking volumes for the day.
- The number of standard deviations from the running mean of the volume, duration, rate and conductivity percentage deviations, and the milking order percentage.

All attributes were scaled to fit within the range of a 16-bit integer, that is $-32768 \ldots 32767$ for signed values or $0 \ldots 65535$ for unsigned quantities. Sequences were classified as positive if the cow was recorded as being in heat on the day in question, or as negative otherwise. If either of the day's milkings was missing from the database, the sequence was discarded. The full dataset contained 13637 sequences in all, with 385 of these being positive. As for Experiment 6, a smaller subset of negative examples was randomly chosen for use in the experiment. This reduced set contained 425 negative examples.

Because the examples were represented as one-dimensional sequences of integers, it was possible to make use of HTK's LP encoding tools to pre-process the data. The sequences were encoded using LP filters of six different orders—2nd, 3rd, 5th, 8th, 10th and 12th order filters were used. A single five-state prototype model was trained and tested on each filtered dataset, following the procedure described above for Experiment 6.

Two further training and test runs were made with a different prototype model and more complex filtering of the raw data. The sequences were encoded using 10th and 12th order filters producing *cepstral* and *cepstral delta* coefficients instead of the simple linear prediction coefficients used above. A new five-state model was trained and tested on these datasets. The initial transition matrices for both models used in this experiment can be found in Appendix B.

Chapter 4

Results and Discussion

This chapter presents a summary of results from the experiments described in the previous chapter, and a discussion of these results in the context of the objectives of the WEKA project and the Dairy Research Corporation. A complete set of results for the experiments can be found in Appendix A. Several problems that arose during the experimental work are discussed further, in particular the problems of small disjuncts [38] and highly skewed class distributions. The following summary of experimental results has been divided into sections relating to the similarity-based and sequence identification experiments respectively, as in the previous chapter.

4.1 Similarity-based Experiments

Many different metrics have been used to quantify the performance of a classification algorithm on a dataset. The most common of these is the simple *classification accuracy*—the percentage of correct classifications made on a set of test data. While it is a popular measure of performance, the classification accuracy is known to have several defects [25, 17]. More robust measures of classifier performance make use of measures such as the size of the descriptions, their complexity and other information-theoretic metrics in addition to the raw accuracy [17, 27].

However, in the context of a real-world classification problem, human users of a discovery system are unlikely to be concerned with theoretical measures of classification complexity and information gain. They will be far more interested in how often the system makes the correct prediction, and how much better it is than predictions made by domain experts. The classification accuracy is a suitable measure to use in this situation since it shows exactly how well a scheme has performed on a particular dataset. The size and complexity of descriptions are less important, provided that the descriptions are general enough to cope with all but the most anomalous test data.

4.1.1 Classification accuracy

Figures 4.1–4.12 show the raw classification accuracy for each dataset used in the similarity-based learning experiments. These figures show the accuracy, on both the training and test data, for C4.5 unpruned trees, pruned trees and rules, and for FOIL rules. The graphs also indicate the *baseline accuracy* of the complete dataset used for each experiment. Baseline accuracy is defined as the percentage of examples that fall in the largest class of the data. It is that accuracy that would be achieved by a naïve learner that simply picks the majority class, and is often used as a standard against which new learning results are compared [15]. The baseline accuracy shown on the plots is that of the *test* data, which in all cases is the complete dataset. The baseline accuracy of many of the training sets is less than this value.

Looking first at Experiments 1–3, in Figures 4.1–4.2; the results from the C4.5 runs with 10% subsets of the data have been averaged and are shown in Figures 4.1, 4.3 and 4.5. The graphs indicate the maximum, minimum and average accuracy values over the ten runs. The C4.5 and FOIL results for the complete dataset are given in Figures 4.2, 4.4 and 4.6. Results are shown for C4.5 unpruned trees, pruned trees and rules, and FOIL rules, for both the training and test data.

As expected the unpruned trees have tended to perform better in training, with the pruned trees and rules showing noticeably better (up to 3%) accuracy

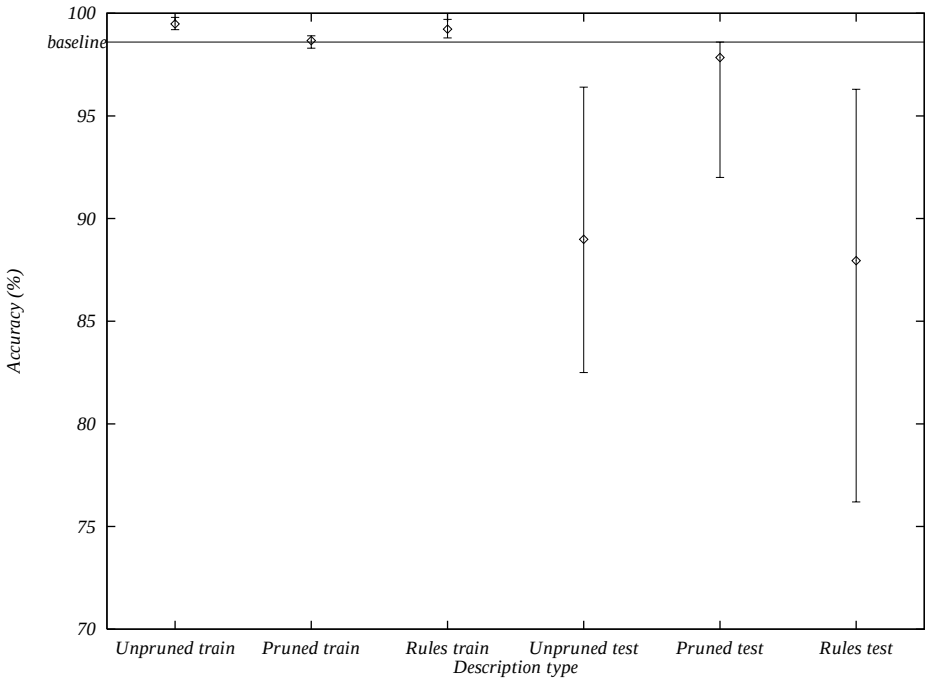

Figure 4.1: Experiment 1 accuracy, C4.5, 10% sets

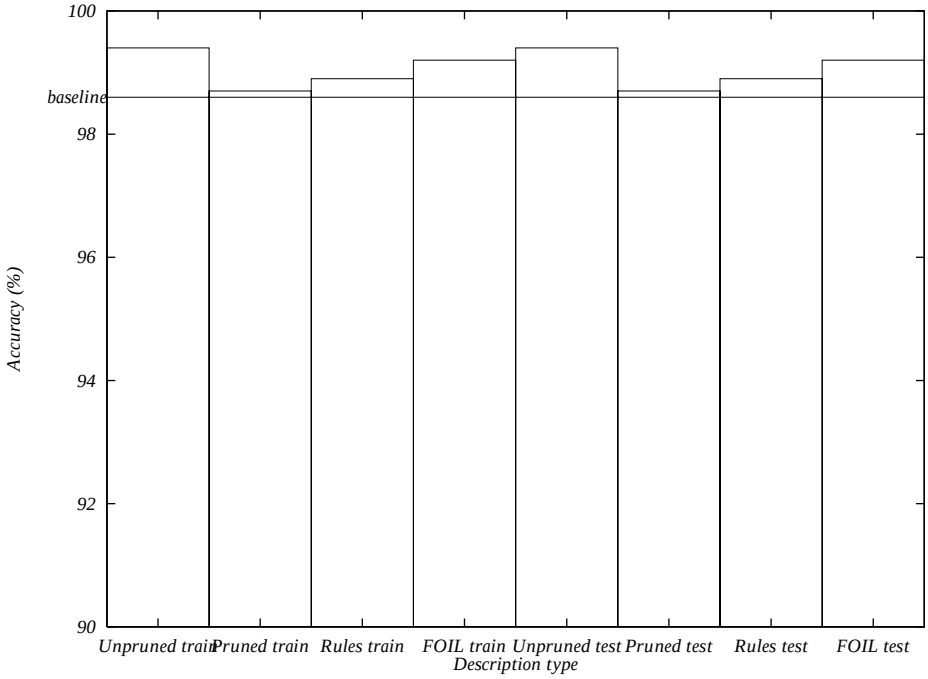

Figure 4.2: Experiment 1 accuracy, C4.5 & FOIL, all data

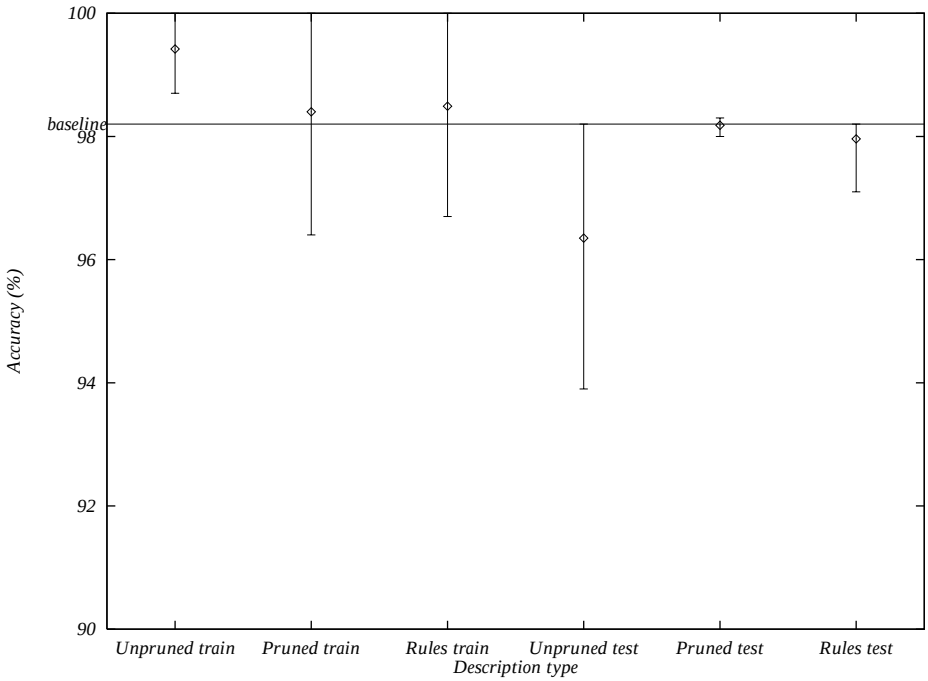

Figure 4.3: Experiment 2 accuracy, C4.5, 10% sets

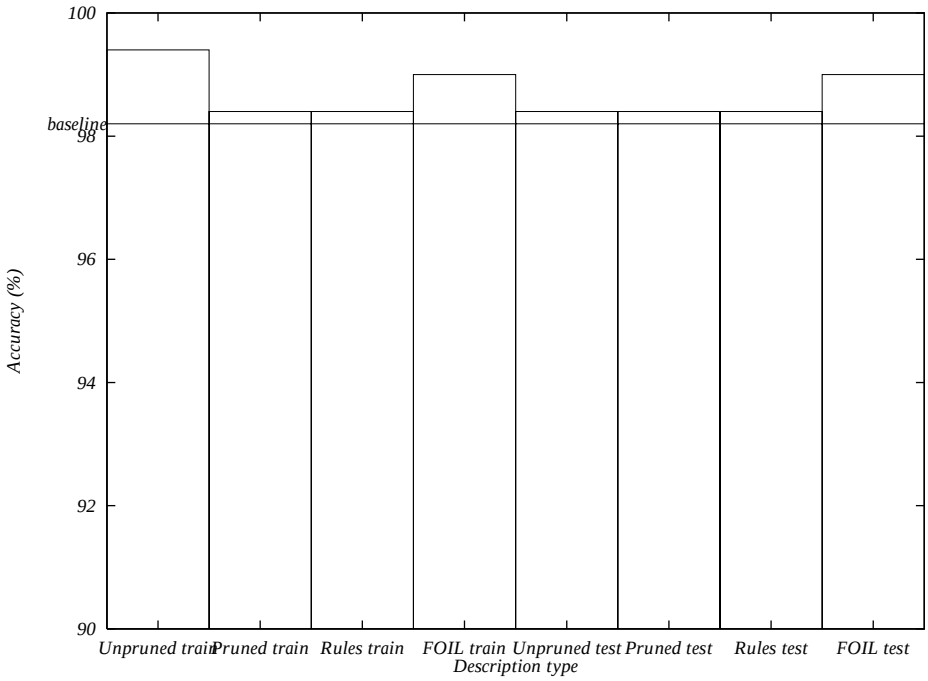

Figure 4.4: Experiment 2 accuracy, C4.5 & FOIL, all data

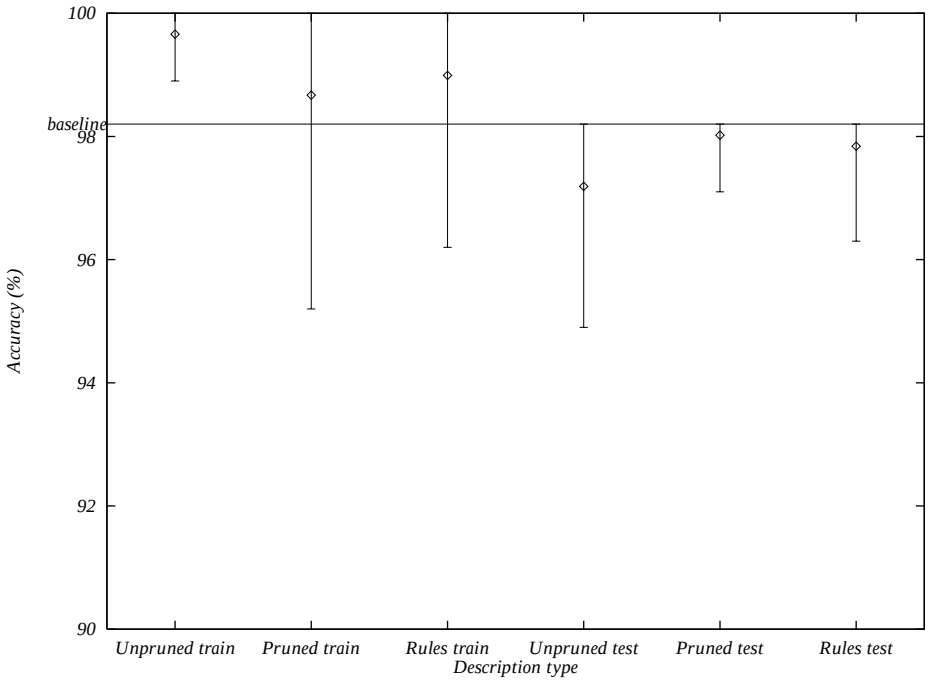

Figure 4.5: Experiment 3 accuracy, C4.5, 10% sets

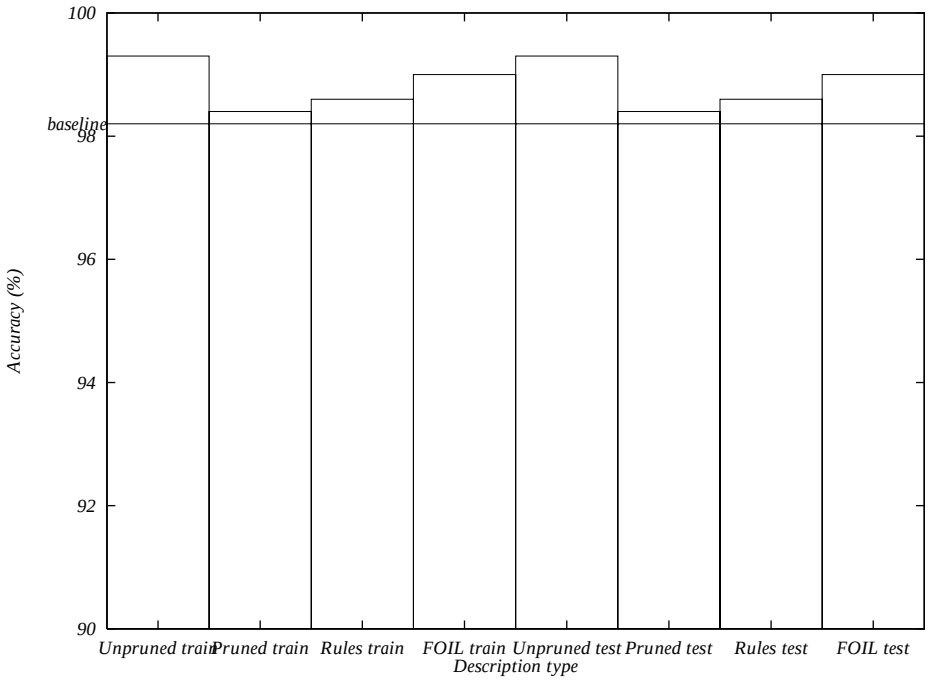

Figure 4.6: Experiment 3 accuracy, C4.5 & FOIL, all data

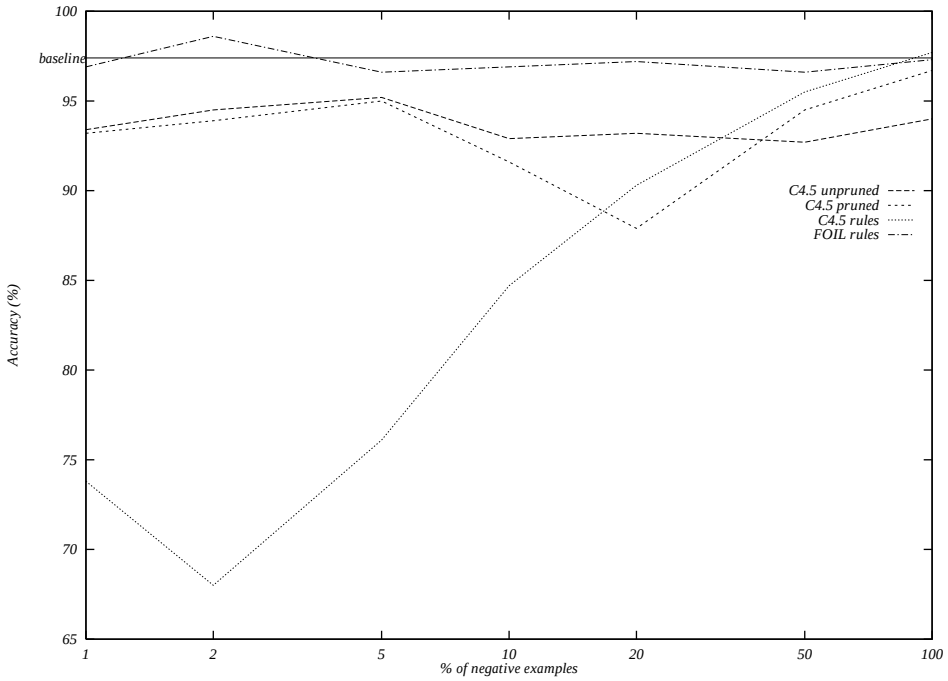

Figure 4.7: Training accuracy for Experiment 4

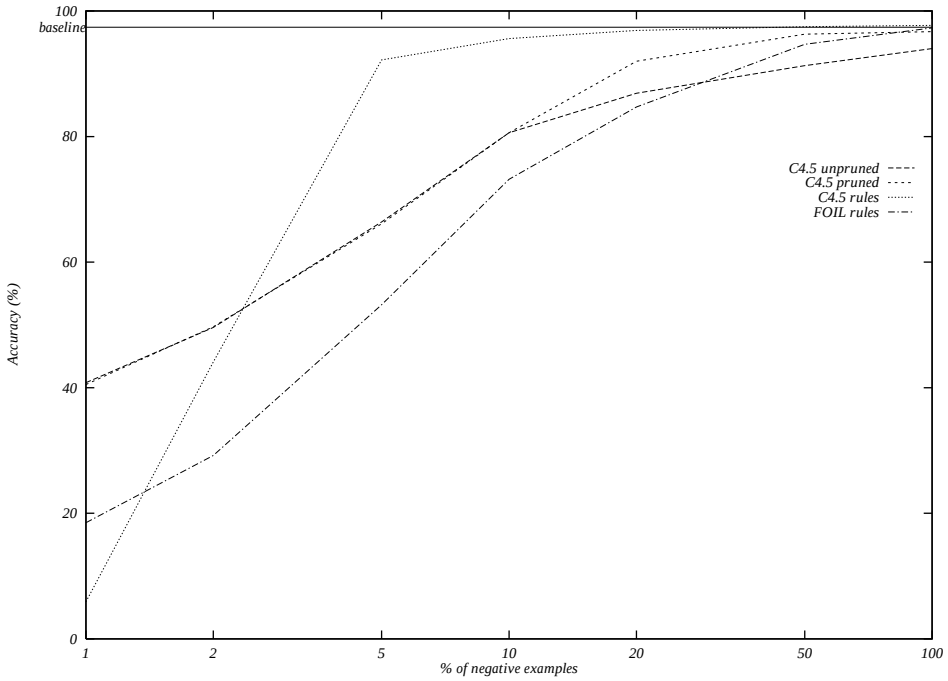

Figure 4.8: Test accuracy for Experiment 4

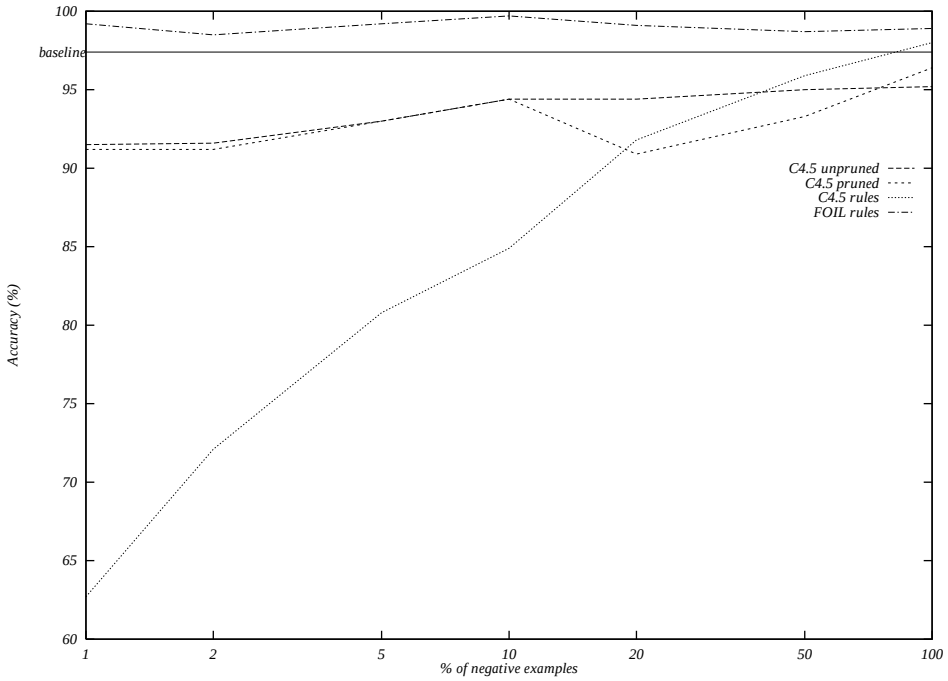

Figure 4.9: Training accuracy for Experiment 5

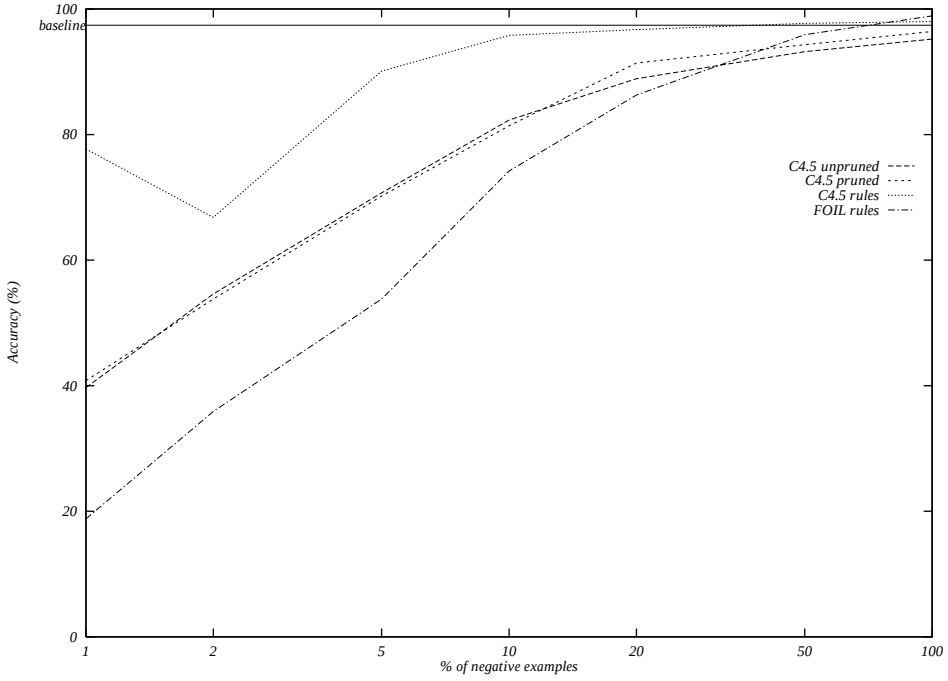

Figure 4.10: Test accuracy for Experiment 5

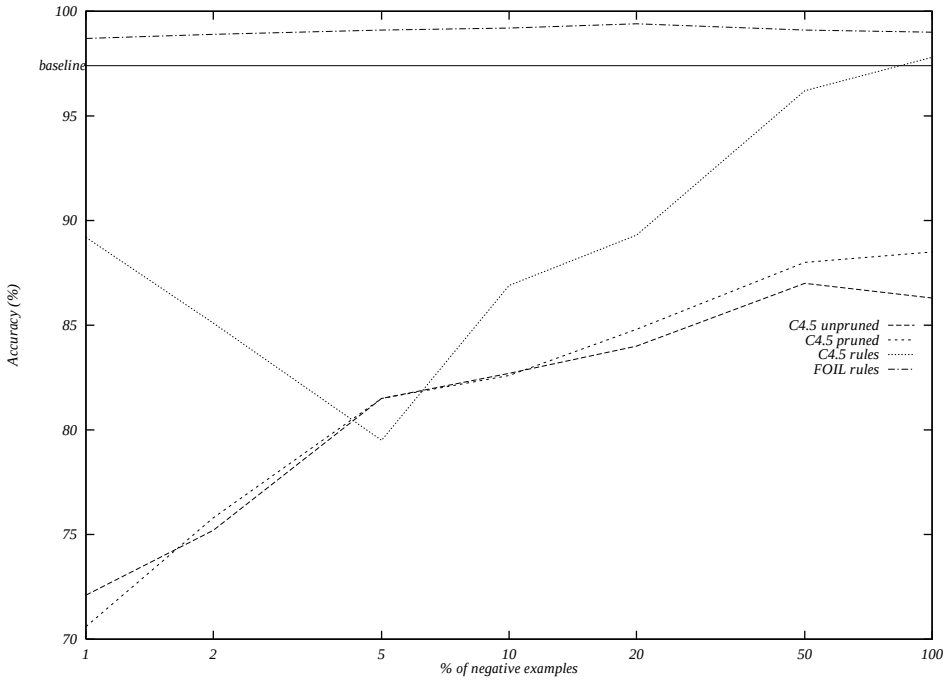

Figure 4.11: Training accuracy for Experiment 5a

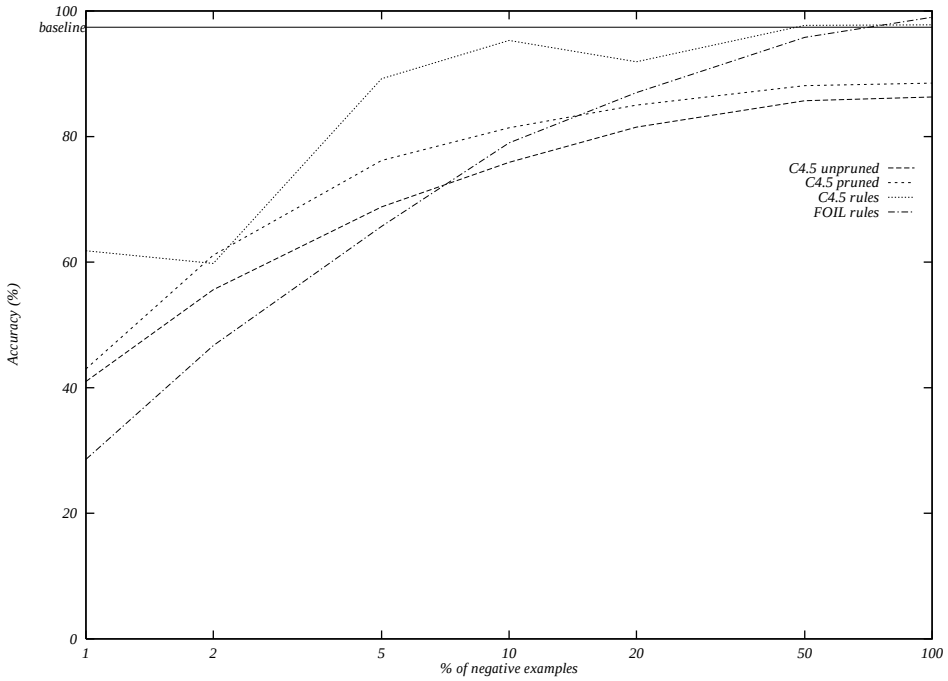

Figure 4.12: Test accuracy for Experiment 5a

on the test data. The most obvious feature of these results, though, is that the test accuracy very rarely rises above the baseline accuracy of the dataset. At first glance this appears to be an extremely poor result, since it means that a scheme that simply classifies all cases as *false* will perform better than either C4.5 or FOIL. However it must be remembered that the baseline accuracy exceeds 95% in all the experiments, making for an *extremely* difficult learning task for any algorithm. For a dataset with a less skewed classification, a classification accuracy greater than 90% would generally be considered an excellent result. Most of the test accuracies, especially for Experiments 2 and 3, are above the 90% mark so in this respect the learning schemes have performed well.

There are a number of possible reasons why C4.5 is unable to achieve the baseline accuracy with this dataset. Firstly, the information necessary to classify the examples this accurately may simply not be present in the data. This situation might occur if there were insufficient examples, if the raw data was particularly noisy or poorly collected, or if an inappropriate attribute set was used for classification. The C4.5 decision tree building algorithm will only generate tests which compare the value of a single attribute with a constant, thus the set of raw and derived attributes must be chosen such that the classification can be described in these terms.

Secondly, the heuristic used by C4.5 to evaluate the quality of tests at decision tree branches may be partly to blame. By default C4.5 uses an information-theoretic heuristic, the *gain ratio criterion* [30]. This is the ratio of the information gained by partitioning a set of examples according to some test, to the *potential* amount of information that could be generated from the same partitioning. Thus, for some test X,

$$\text{gain ratio}(X) = \text{info gain}(X)/\text{partition info}(X) \tag{4.1}$$

which expresses the proportion of information generated by the partition that is useful for classification purposes. The gain ratio criterion selects a test that maximises the value of this ratio. According to Quinlan [30], the gain ratio criterion usually gives a better choice of test than the *gain criterion* used by ID3. However, Mingers [26] expresses some concern about the tendency of

the gain ratio criterion to favour highly skewed partitionings of the examples. Often when such unequal "splits" occur, the small group—possibly consisting of only two or three examples—is purely of one class, and the split will not be divided any further. Quinlan argues that the tendency of the gain ratio criterion to form such splits is desirable, because it generally leads to smaller decision trees. Statistically however, these small group are most likely to be chance occurrences that are unreliable for predicting test cases. In the statistical sense it is better to have larger groups, possibly containing a few examples from different classes.

The heavily skewed distribution of the herd milking database and the bias inherent in the gain ratio criterion means that there is ample opportunity for small splits to occur in decision trees induced from this data. The C4.5 system does offer two possible solutions to the problem:

1. The decision tree inducer can use the older gain criterion instead of the gain ratio measure.

2. The sizes of the partitions produced by a test can be restricted, so that each outcome of the partitioning must contain at least a specified minimum number of examples. By default the minimum partition size is two examples. Increasing this value would prevent C4.5 from considering many potential small splits in the decision tree.

Experiments 4–5a (Figures 4.7–4.8) were designed to reduce the problem of skewed classifications by using a smaller proportion of the negative examples for training. The implicit assumption was that a smaller set of examples would be sufficient to induce accurate descriptions while avoiding the problems encountered in the earlier experiments. The figures show the training and test accuracy of the C4.5 unpruned trees, pruned trees and rules, and the FOIL rules, as a function of the percentage of negative examples used. As shown by the test accuracy graphs for these experiments (Figures 4.8, 4.10 and 4.12), this approach has been partially successful.

Again as expected, the C4.5 pruned trees and rules have performed better in testing than the less general unpruned trees. It is somewhat surprising that the rules generated by FOIL have not performed particularly well on the test data with respect to the C4.5 descriptions. This can probably be accounted for by the fact that FOIL does not do any pruning of its rulesets, so testing is done using unpruned rules and will tend to perform badly whenever there is noise present in the test data. FOIL consistently out-performed C4.5 on the training runs, in many cases giving training accuracies significantly higher than the baseline accuracy.

As with Experiments 1–3, very few of the test accuracies for Experiments 4–5a have exceeded the baseline accuracy. In particular the descriptions trained using less than 20% of the negative examples have performed quite badly, often producing test accuracies of less than 70%. However, most of the training sets using 20% or more of the negative examples have generated descriptions with accuracy above 80%. Many of the pruned tree and rule accuracies exceed 90%. It should also be noted that almost all of the training runs have performed better than baseline accuracy of their respective *training* sets. The relatively poor performance of the smaller datasets—those using less than 20% of the negative examples—indicates that there is significant noise and natural variation present in the examples, so that a relatively large training set is required to cover all of the anomalous situations that occur.

4.1.2 Types of classification errors

It is also interesting to examine the *types* of classification errors being made by the various descriptions. Figures 4.13–4.18 show the percentages of positive and negative test examples misclassified for Experiments 3, 5 and 5a. As for the classification accuracy graphs, separate curves are given for unpruned trees, pruned trees, C4.5 rules and FOIL rules.

The majority of misclassifications made by the trees and rules of Experiment 3 are “false negatives”—positive examples misclassified as negative. The

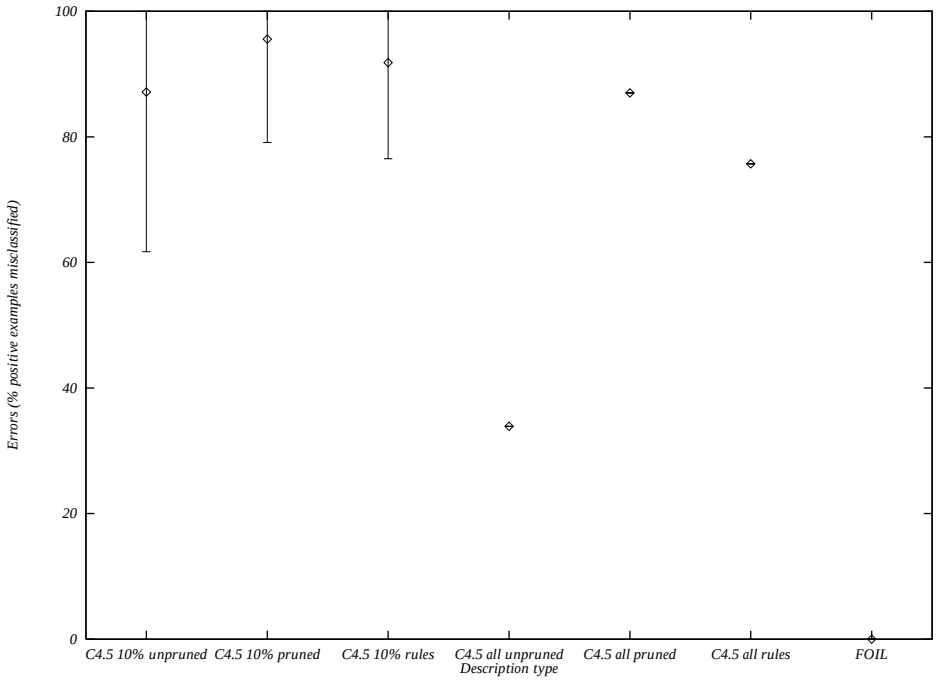

Figure 4.13: Experiment 3 T⇒F misclassifications on test data

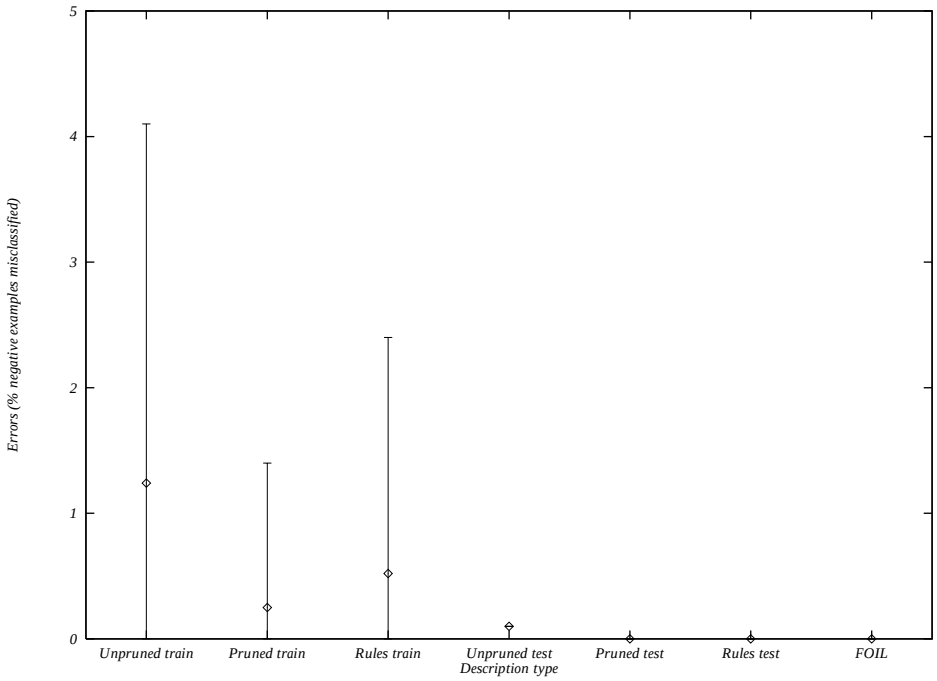

Figure 4.14: Experiment 3 F⇒T misclassifications on test data

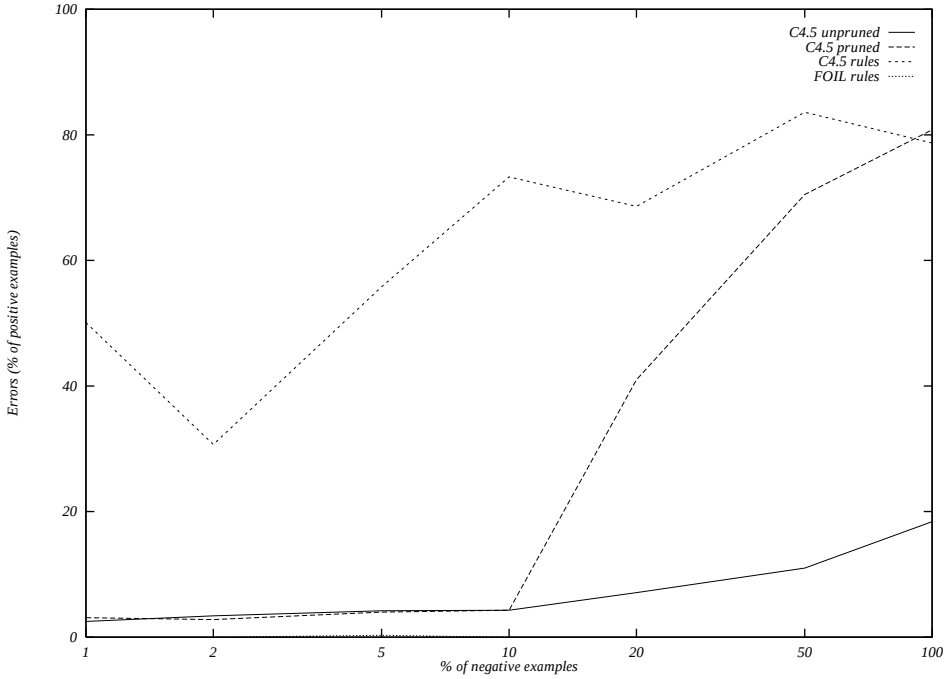

Figure 4.15: Experiment 5 T⇒F misclassifications on test data

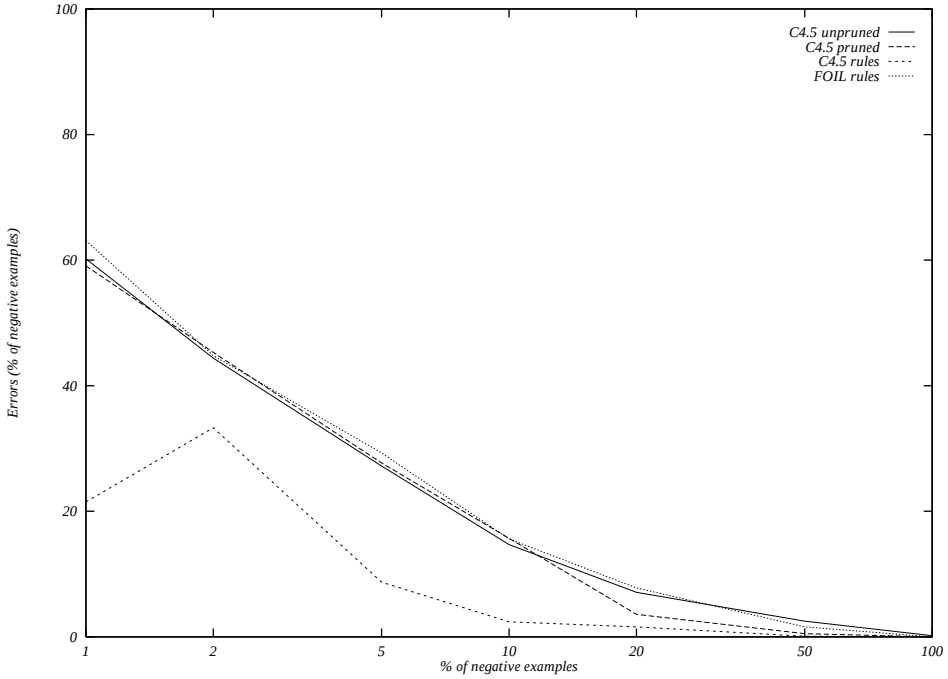

Figure 4.16: Experiment 5 F⇒T misclassifications on test data

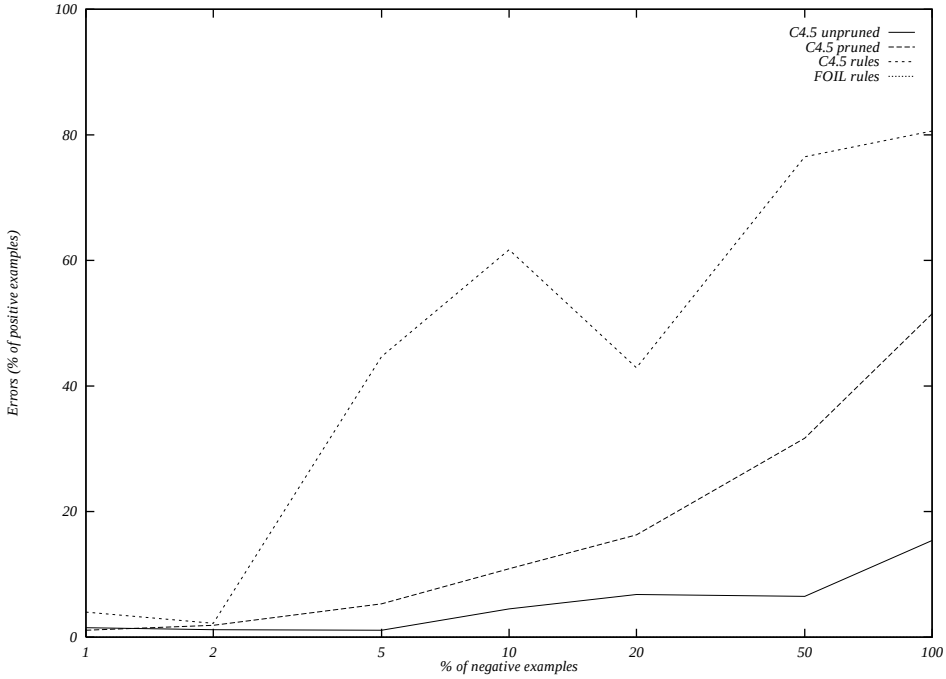

Figure 4.17: Experiment 5a T⇒F misclassifications on test data

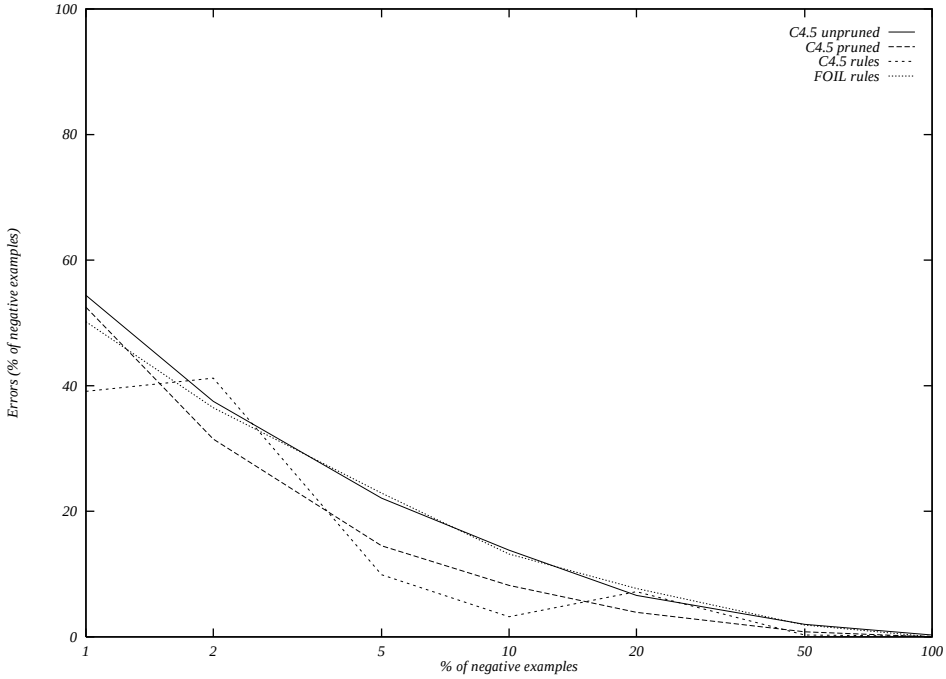

Figure 4.18: Experiment 5a F⇒T misclassifications on test data

rate of false positive classifications is less than 4% for nearly all of the test runs. This is indicative of the small splits problem discussed above—small clusters of positive examples give good performance in training but lack any real predictive power on the test data. FOIL does not suffer from this problem and has low misclassification rates for both positive and negative examples. This is not especially useful however since for this experiment FOIL was trained on the entire dataset, so a low test error rate is to be expected for the unpruned FOIL rules.

The misclassification results for Experiments 5 and 5a are quite similar: as the proportion of negative examples in the training set increases, the percentage of false negative classifications goes up as the percentage of false positive decreases. This is particularly noticeable with the C4.5 pruned trees and rules.

FOIL has performed particularly well on these experiments, with low misclassification rates for both positive and negative examples. The number of false positive classifications generated by the FOIL rules drops significantly as the proportion of negative examples approaches the 10–20% level. This is further evidence that below this level, the amount of random variation present in the negative examples prevents an accurate description being induced from the training data.

The proportion of false negative classifications made by C4.5 rises as the proportion of negative examples—and thus the classification skew—increases. This is another example of the small splits problem. The effect is less severe for the unpruned rules because the small split decisions are at least able to correctly classify the examples that caused the split to occur in the first place. Many of the small splits will disappear from the tree during pruning, with the small groups of positive examples from these decisions then incorrectly classified as negative.

Ting [38] has developed a composite learning scheme that partly alleviates the problem of small splits or *small disjuncts*. His method uses an instance-based learner [1] to classify test cases belonging to a small disjunct, otherwise the C4.5 decision tree is used. Both learners are trained in parallel on the same training

set. The advantage of this system is that the specific instances that caused the small disjunct are used to classify test cases, instead of relying on generalised—and probably inaccurate—rules. The performance of large disjuncts is not affected since these are still classified using the C4.5 decision tree. On a selection of datasets from the UCI machine learning database repository, the composite learner performed on average 2.5% better in testing than C4.5 alone.

4.2 Sequence Identification Experiments

The Viterbi recognition package included with HTK produces an overall recognition accuracy figure and a confusion matrix, for each testing run. It is also possible to generate a list of the individual cases that were misclassified. Figures 4.19–4.22 show the test accuracy and the positive and negative misclassification results for Experiments 6 and 7. Figures 4.19 and 4.20 also indicate the baseline accuracy of the respective test sets. The baseline accuracy for these experiments is significantly lower than for Experiments 1–5a because the models were trained and tested on a subset of the data having an approximately uniform class distribution. This use of a reduced dataset was necessary due to the limit on the number of training sequences imposed by HTK.

Although the test accuracy results in Figures 4.19 and 4.20 consistently exceed the baseline accuracy for the test set used in the experiment, they are significantly below the baseline accuracy of the whole dataset. The baseline accuracy for both experiments was close to 50%, with the majority of the test results lying in the 55–60% range. A baseline accuracy of 50% is not particularly useful in the context of a two-class problem, since a learner could achieve the same level of accuracy simply by making random guesses. Thus the Markov models have performed better than random chance, but only by a narrow margin. A classification accuracy of less than 60% is unlikely to be considered a “good” result under any circumstances.

There are several possible explanations for the poor performance of the Markov modelling technique on this dataset. Firstly, it is clear that there is

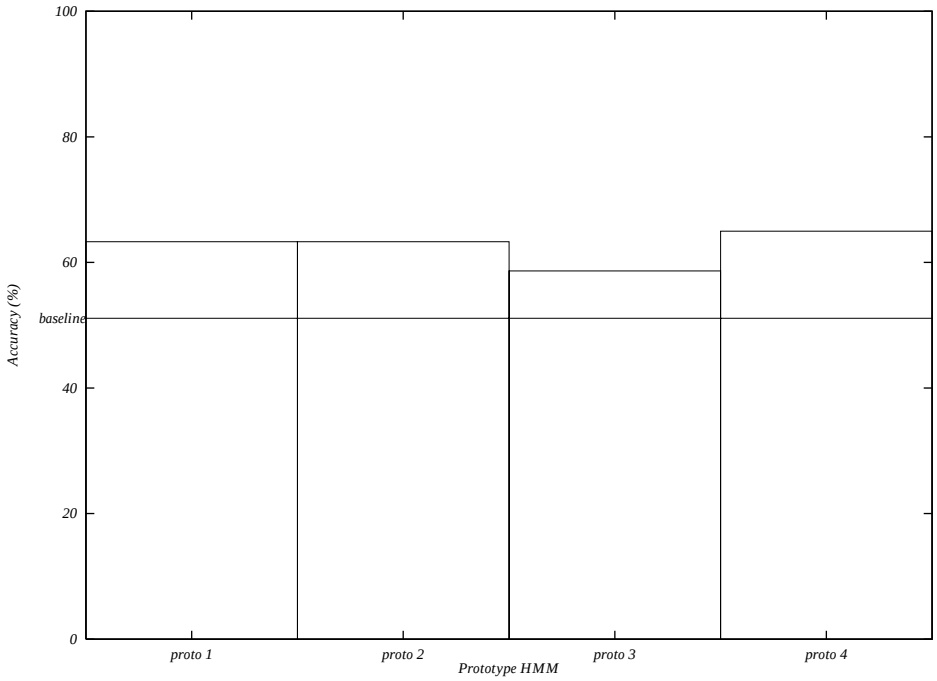

Figure 4.19: Experiment 6 test accuracy

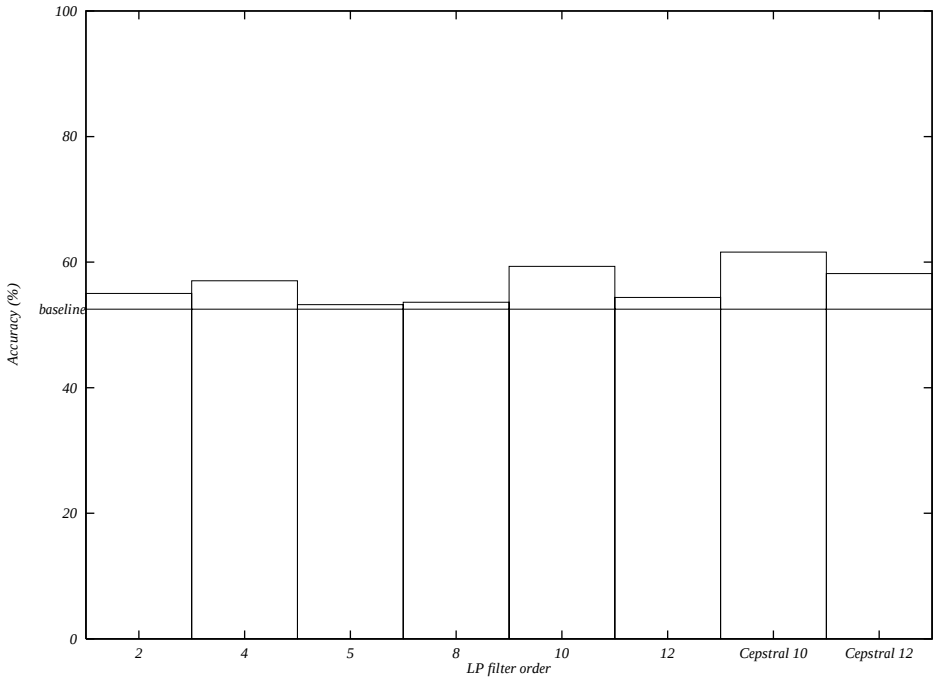

Figure 4.20: Experiment 7 test accuracy

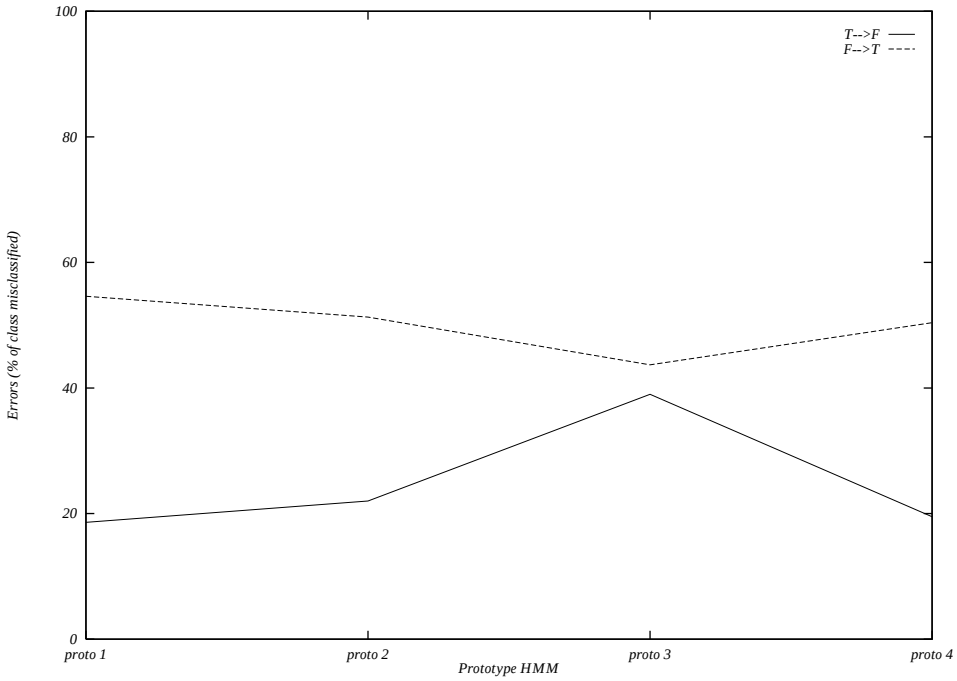

Figure 4.21: Experiment 6 misclassifications

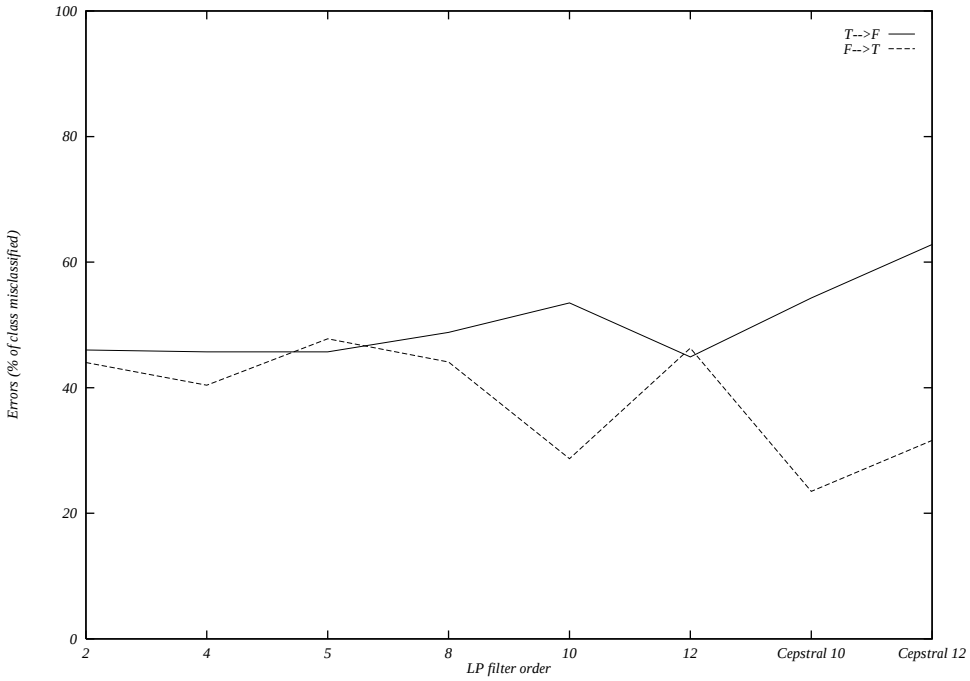

Figure 4.22: Experiment 7 misclassifications

a significant amount of noise and random variation present in the milking database. The similarity based schemes discussed above performed poorly on training sets containing less than 10–20% of the available negative examples. The training sets in the hidden Markov model experiments used approximately 2% of the negative examples, so it is possible that there was simply not enough "clean" data present for HTK to build an accurate model. The model parameter estimation process is statistically-based, so the presence of noise will tend to "blur" any distinctions between the classes, leading to an inaccurate model.

Secondly, it is probable that many of the cows in the database do not show any obvious indication of the times they are in heat. Robert Sherlock from the DRC has estimated that "in heat" events can be easily identified by visual inspection for as little as 20% of the herd. A further unknown proportion of the herd—estimated at up to 50%—will exhibit similar but less obvious patterns that are hard to identify visually. A similarity- or statistically-based learning scheme should be able to correctly classify these groups, given sufficient training examples from both sets. The remainder of the herd is not expected to show any pattern similar to these two groups. Of course it is possible that the data for the remaining cows *will* contain different, non-obvious patterns that could be used for classification.

While a similarity-based learner would be able to classify the non-obvious examples *separately*, using whatever similarities did exist between the instances of each class, a statistically-based system cannot do this. HTK will attempt to build a single model that accurately describes *all* of the training examples for a particular class. The examples from cows that do not show any recognisable pattern contribute very little information about the class, but the model will still attempt to account for them. This behaviour leads to both models trying to classify a group of quite similar examples. Consequently, the models will perform poorly when an instance from this group is encountered during testing. Either model could give a better recognition score in this case, so these examples will effectively be classified randomly.

Finally, the initial models, attribute sets or observation vectors used in the

experiments may have been inappropriately chosen. A variety of different models were used in the experiments, with no significant improvements in classification accuracy for any particular model. Thus it is unlikely that the choice of model is to blame for the poor testing performance. It is far more likely that the choice of attribute set or the way that the examples were presented to the modeler is responsible.

HTK expects its raw input data to be a time-series of samples of a single variable. It is not particularly set up to deal with the multi-dimensional data used in Experiment 6 and 7. Even after the data had been manipulated into into a form that was acceptable to HTK, it was necessary to in effect “lie” to HTK about the source format and encoding of the data. This is because the HTK interface assumes that it is dealing with speech signals and cannot be changed from this point of view.

The sequences of multi-dimensional observations used in Experiment 6 were presented to HTK as though they were LP encoded versions of single-dimensional raw data. HTK will attempt to make use of the supposed relationship between the elements of each observation vector, although these are in fact potentially independent variables. In Experiment 7 the attribute values from each example were normalised and presented to HTK as a single-dimensional raw sequence. HTK could be justified in assuming a temporal relationship between the observations that did not in fact exist. This discrepancy between the form of observations expected by HTK and the data available from the herd milking database is very likely to lead to poor classification results, as observed in the experiments. Some possible solutions to this problem will be discussed in the following chapter.

Chapter 5

Conclusions and Future Work

The goals of this project were: firstly, to investigate the use of machine learning and knowledge discovery techniques and algorithms for the analysis of real-world time-series data; and secondly, to apply these techniques to the DRC herd milking database to produce useful and relevant descriptions. Two different approaches to analysing time-series data have been investigated:

1. Using conventional similarity-based learning techniques to describe the structure of the time-series database, in particular C4.5 and FOIL.
2. Using sequence identification techniques from signal processing applications such as speech recognition. A hidden Markov modelling system was used to learn statistically-based descriptions of the time-series data.

The database used in this study contains real-world data, and has features that are not often encountered with the datasets commonly used in the machine learning literature:

- The class distribution is highly skewed—approximately 98% of the records in the database describe negative examples.
- There is a significant amount of random noise present in the data, as well as missing values where the necessary data has not been collected.

- The database is relatively large, containing approximately 38,000 individual records.

Similarity-based learning schemes have no concept of time, so a significant amount of pre-processing was required to present the database in a form that made the time dimension implicitly available to the learner. In the case of the hidden Markov modeller, the time-series can be explicitly represented. However, pre-processing was still required to generate suitable sets of attributes and observations for presentation to the modeller.

C4.5 is able to deal with features of real-world data such as noise and missing values, and in general has performed well on both training and test datasets. The major source of error lies in the extreme skewness of the class distribution. C4.5's gain-ratio criterion reacts badly to this type of distribution and tends to build decision trees containing many small disjuncts, that perform poorly in testing. This type of highly skewed data has caused similar problems in other cases studies undertaken by the WEKA projects [40]. Clearly this is a common and significant problem when working with real-world datasets, and a methodology needs to be developed to address it. For the particular case of C4.5 it is possible that the parameters of the tree induction algorithm could be optimised on a case-by-case basis. In general however a more fundamental solution is required. As mentioned previously Ting [38] has obtained useful results with a composite scheme using a combination of C4.5 and an instance-based learner. The success of this composite learning technique makes it a good candidate for further research in this area.

The rulesets generated by FOIL performed particularly well on training data. This is because FOIL is able induce full first-order logical relations from the examples—thus it is able to express the implicit time-series relationship between variables within an instance. FOIL rules consistently performed poorly in testing however, primarily because no pruning of the ruleset was performed. First-order—and higher—schemes such as FOIL have great potential for learning time-series relations. Before this potential can be realised these schemes need to incorporate more of the advanced features now commonplace in zero-

order decision-tree learners, especially pruning and other forms of ruleset generalisation and optimisation.

Hidden Markov modelling is based around building a parameterised statistical model of the data. Thus, the sequences used for training need to be statistically "close" in order to generate a model that will only recognise sequences that are similar to the training data. Unfortunately this is not true for the DRC herd milking database. The data effectively contains two distinct distributions of cows—those that exhibit a recognisable pattern in the time-series and those that do not. This dual distribution cannot be accurately described by a single set of models. The end result is the poor classification accuracy seen in Experiments 6 and 7.

The HMM technique has already proven to be very effective with speech signals—which are just simple single-dimensional time-series data. The milking data differs in that it is multi-dimensional, and the sequences used are significantly shorter than would be encountered in a typical speech processing application. The milking data also has the advantage of being highly periodic. There would seem to be no obvious reason why the HMM method should not be capable of performing well in application domains other than speech, given a homogeneous set of training examples and a suitable prototype model.

5.1 Future Work

Research is continuing in a number of areas related to the DRC milking database and to the general problem of time-series data analysis. Firstly, the HTK Markov modelling system is to be integrated into the WEKA workbench. This involves:

- Construction of a graphical user interface to the HTK tools.
- Extensions to the ARFF file format to directly handle sequence data.
- Modifications to the HTK code to allow larger training sets and the ability to LP encode multi-dimensional data.

Integration into the workbench gives access to various WEKA tools such as the attribute editor and experiment editor, hopefully allowing new experiments to be designed much more quickly, with automatic processing and collation of results.

Secondly, two new herd milking databases have recently been received from the DRC. The first of these contains data from the 1994–95 milking season for the same DRC herd as the 1993–94 database used in this study. The makeup of the DRC herd has not changed significantly between seasons so there is expected to be a high correlation of the characteristics of individual animals across the two databases. The second new dataset comes from a production farm and contains data for the 1994–95 season. Initially it is planned to repeat some of the experiments described in this report with the new datasets, to verify the results obtained with the original DRC database. All three datasets will be used in the new experiments described below. It should also be possible to combine the data for cows appearing in both of the DRC herd datasets into a single large dataset. This will allow experiments to be run on larger amounts of training and test data.

Finally, based on the results obtained in this study we plan to conduct further experiments, particularly in the area of hidden Markov modelling. Several new approaches are under consideration, including:

- Building models based only on the subset of the herd that exhibits a relatively easy to identify pattern. This subset will initially be chosen by visual inspection of the data. Although this method cannot expect to produce models that accurately classify all possible test cases, it should eliminate a large amount of the error encountered in Experiments 6 and 7. Models that correctly classify a proportion of the herd are far more useful than models that perform poorly on the entire herd.
- As a limiting case of the method above, separate models could be constructed for each individual cow. In theory this should produce very accurate models, at the cost of some generality—new models would have

to build over a period of time whenever a new cow joined the herd. Models generated for one herd would also not be applicable to any other herd. The success of this approach would at least show that Markov modelling is a viable technique for this application.

- Some investigation of different prototype models, derived attribute sets and observation vectors should also be carried out using all three datasets.

We also plan to test the effects of modifying the C4.5 induction parameters—in particular those affecting the operation of the gain-ratio criterion—and possibly apply a pruning algorithm to rulesets generated by FOIL in order to improve their test performance.

Bibliography

[1] D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.

[2] Lalit R. Bahl, Frederick Jelinek, and Robert L. Mercer. A maximum likelyhood approach to continuous speech recognition. In Alex Waibel and Kai-Fu Lee, editors, *Readings in Speech Recognition*, pages 308–319. Morgan Kaufmann Publishers, Inc., San Mateo, California, 1990.

[3] L. E. Baum. An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. *Inequalities*, 3:1–8, 1972.

[4] Donald J. Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In Usama M. Fayyad and Ramasamy Uthurusamy, editors, *Proceedings of KDD-94: AAAI-94 Workshop on Knowledge Discovery in Databases*, pages 359–370, Seattle, Washington, July–August 1994. American Association for Artificial Intelligence.

[5] Ronald J. Brachman and Tej Anand. The process of knowledge discovery in databases: A first sketch. In Usama M. Fayyad and Ramasamy Uthurusamy, editors, *Proceedings of KDD-94: AAAI-94 Workshop on Knowledge Discovery in Databases*, pages 1–11, Seattle, Washington, July–August 1994. American Association for Artificial Intelligence.

[6] J. Cendrowska. PRISM: An algorithm for inducing modular rules. *International Journal of Man-Machine Studies*, 27(4):349–370, 1987.

[7] P. Cheeseman, J. Kelly, M. Self, J. Stutz, W. Taylor, and D. Freeman. AutoClass: A Bayesian classification system. In *Proceedings of the Fifth International Conference on Machine Learning*, pages 54–64, San Mateo, California, 1988. Morgan Kaufmann Publishers, Inc.

[8] D. Fisher. Knowledge acquisition via incremental concept clustering. *Machine Learning*, 2:139–172, 1987.

[9] Brian R. Gaines. Exception dags as knowledge structures. In Usama M. Fayyad and Ramasamy Uthurusamy, editors, *Proceedings of KDD-94: AAAI-94 Workshop on Knowledge Discovery in Databases*, pages 13–24, Seattle, Washington, July–August 1994. American Association for Artificial Intelligence.

[10] Stephen R. Garner. ARFF—the WEKA dataset format. World Wide Web hypertext document at *http://www.cs.waikato.ac.nz/~ml/workbench/arff.html*, 1994.

[11] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publishing Company Inc., Reading, Massachusetts, 1989.

[12] S. Hansen and M. Bauer. Conceptual clustering, categorization and polymorphy. *Machine Learning*, 3:343–372, 1989.

[13] Geoffrey Holmes, Andrew Donkin, and Ian H. Witten. WEKA: A machine learning workbench. Working Paper 94/9, Department of Computer Science, University of Waikato, Hamilton, New Zealand, July 1994.

[14] Marcel Holsheimer and Arno Siebes. Data mining: The search for knowledge in databases. Technical Report CS-R9406, CWI, Amsterdam, The Netherlands, 1994.

[15] Robert C. Holte. Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11:63–91, 1993.

[16] Willi Klosgen and Jan M. Zytkow. Machine discovery terminology. In Usama M. Fayyad and Ramasamy Uthurusamy, editors, *Proceedings of*

KDD-94: AAAI-94 Workshop on Knowledge Discovery in Databases, pages 463–473, Seattle, Washington, July–August 1994. American Association for Artificial Intelligence.

[17] I. Kononenko and I. Bratko. Information-based evaluation criterion for classifier's performance. *Machine Learning*, 6:67–80, 1991.

[18] M. Lebowitz. Experiments with incremental concept formation: UNIMEM. *Machine Learning*, 2:103–138, 1987.

[19] Kai-Fu Lee. *Automatic Speech Recognition: The Development of the SPHINX System*, chapter 2, pages 17–43. Kluwer Academic Publishing, Boston, Massachusetts, 1989.

[20] Stephen E. Levinson, Lawrence R. Rabiner, and M. M. Sondhi. An introduction to the application of the theory of probabilistic functions on a Markov process to automatic speech recognition. *The Bell System Technical Journal*, 62(4), April 1983.

[21] Robert J. McQueen, Stephen R. Garner, Craig G. Nevill-Manning, and Ian H. Witten. Applying machine learning to agricultural data. Working Paper 94/13, Department of Computer Science, University of Waikato, Hamilton, New Zealand, July 1994.

[22] Robert J. McQueen, Donna Neal, Rhys DeWar, and Stephen R. Garner. Preparing and processing relational data through the WEKA machine learning workbench. Working paper, Department of Computer Science, University of Waikato, Hamilton, New Zealand, 1994.

[23] R. Michalski and R. Stepp. Learning from observation: Conceptual clustering. In R. Michalski, R. Carbonell, and T. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*, pages 331–363. Tioga Publishing Company, Palo Alto, California, 1983.

[24] D. Michie. Methodologies from machine learning in data analysis and software. *The Computer Journal*, 34(6):559–565, 1991.

[25] John Mingers. An empirical comparison of pruning methods for decision-tree induction. *Machine Learning*, 4:227–243, 1989.

[26] John Mingers. An empirical comparison of selection measures for decision-tree induction. *Machine Learning*, 3:319–342, March 1989.

[27] Trevor J. Monk, R. Scott Mitchell, Lloyd A. Smith, and Geoffrey Holmes. Geometric comparison of classifications and rule sets. In Usama M. Fayyad and Ramasamy Uthurusamy, editors, *Proceedings of KDD-94: AAAI-94 Workshop on Knowledge Discovery in Databases*, pages 395–406, Seattle, Washington, July–August 1994. American Association for Artificial Intelligence.

[28] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[29] J. R. Quinlan. Learning logical relation from definitions. *Machine Learning*, 5:239–266, 1990.

[30] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, Inc., San Mateo, California, 1992.

[31] Lawrence R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. In Alex Waibel and Kai-Fu Lee, editors, *Readings in Speech Recognition*, pages 267–296. Morgan Kaufmann Publishers, Inc., San Mateo, California, 1990.

[32] Lawrence R. Rabiner and B-H Juang. *Fundamentals of Speech Recognition*, chapter 6, pages 321–389. Prentice-Hall, New York, 1993.

[33] Lawrence R. Rabiner and Stephen E. Levinson. Isolated and connected word recognition—theory and selected applications. In Alex Waibel and Kai-Fu Lee, editors, *Readings in Speech Recognition*, pages 115–153. Morgan Kaufmann Publishers, Inc., San Mateo, California, 1990.

[34] Lawrence R. Rabiner, Jay G. Wilpon, and Frank K. Soong. High performance connected digit recognition using hidden Markov models. In Alex Waibel and Kai-Fu Lee, editors, *Readings in Speech Recognition*, pages 320–331. Morgan Kaufmann Publishers, Inc., San Mateo, California, 1990.

[35] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. In Alex Waibel and Kai-Fu Lee, editors, *Readings in Speech Recognition*, pages 159–165. Morgan Kaufmann Publishers, Inc., San Mateo, California, 1990.

[36] Robert A. Sherlock. Private e-mail communication, February 1995.

[37] Padhraic Smyth, Michael Burl, Usama Fayyad, and Pietro Perona. Knowledge discovery in large image databases: Dealing with uncertainties in ground truth. In Usama M. Fayyad and Ramasamy Uthurusamy, editors, *Proceedings of KDD-94: AAAI-94 Workshop on Knowledge Discovery in Databases*, pages 109–120, Seattle, Washington, July–August 1994. American Association for Artificial Intelligence.

[38] Kai Ming Ting. The problem of small disjuncts: its remedy in decision trees. In Renée Elio, editor, *Proceedings of the Tenth Canadian Conference on Artificial Intelligence*, pages 91–97, Banff, Alberta, May 1994. Canadian Society for Computational Studies of Intelligence.

[39] Paul E. Utgoff. Incremental induction of decision trees. *Machine Learning*, 4:161–186, 1989.

[40] Rhys De War and Donna Liane Neal. WEKA machine learning project: Cow culling. Working Paper 94/12, Department of Computer Science, University of Waikato, Hamilton, New Zealand, 1994.

[41] Steve J. Young. *HTK: Hidden Markov Model Toolkit V1.2 Installation Guide*. Cambridge University Engineering Department Speech Group, Cambridge, England, December 1990.

[42] Steve J. Young. *HTK: Hidden Markov Model Toolkit V1.2 Reference Manual*. Cambridge University Engineering Department Speech Group, Cambridge, England, December 1990.

Appendix A

Experimental Results

This appendix presents complete results for all of the experiments described previously.

A.1 Similarity-Based Learning Experiments

The results presented in this section were taken directly from the output of the WEKA Prolog evaluator. A set of nine tables of results is provided for each of the similarity-based learning experiments, containing the following information:

Example counts. The total number of examples in the *full* dataset for each experiment, and the numbers of positive and negative examples in this set. For Experiment 1, the *yes* and *maybe* classes are combined to form the positive examples.

Baseline accuracy. The proportion of examples falling in the largest class of the dataset. Baseline accuracy is often used as a basis for comparing the performance of learning schemes [15, 17]. The baseline accuracy figures given below are for the complete dataset used in each experiment.

Dataset size. The total number of examples in the dataset used for each trial.

Ruleset size. The number of rules or decision tree leaves in the descriptions generated by the learning scheme.

Accuracy. The overall accuracy of the description with respect to the dataset, expressed as the percentage of examples classified correctly.

Incorrect classifications. The percentage of examples classified incorrectly by the description.

Unclassified examples. The percentage of examples which were not classified at all by the description.

Multiply classified examples. The percentage of examples classified by more than one rule or leaf.

$T \Rightarrow F$ **misclassifications.** The number of positive examples in the dataset which were misclassified as negative. This is expressed as a percentage of the total number of positive examples, including multiple classifications—examples classified as both positive and negative.

$F \Rightarrow T$ **misclassifications.** The number of negative examples in the dataset which were misclassified as positive. Expressed as a percentage of the total negative examples.

With the exception of baseline accuracy and example counts, each of these variables is presented for both the training and test sets used in each trial.

For Experiments 1–3, the trials identified as "C4.5 set 1"–"C4.5 set 10" correspond to the ten data subsets used in the ten-way cross-validation experiments. Each set consists of 10% of the examples from the full dataset, with no overlap between subsets.

The Experiment 4–5a results show a percentage figure beside the name of the learner used in each trial. This number indicates the percentage of the available negative examples included in the training set. In all cases testing was done using the whole, 100% dataset.

A.1.1 Experiment 1

Examples: 38360; **Positive:** 536, **Negative:** 37824

Baseline accuracy: 98.6%

	Training Size	Ruleset Size	Training Accuracy%	Test Accuracy%
C4.5 (all)	38360	505	99.4	99.4
C4.5 set 1	3836	40	99.5	96.4
C4.5 set 2	3836	27	99.6	88.0
C4.5 set 3	3836	27	99.5	96.8
C4.5 set 4	3836	56	99.5	89.5
C4.5 set 5	3836	53	99.6	84.4
C4.5 set 6	3836	54	99.3	93.8
C4.5 set 7	3836	53	99.5	82.5
C4.5 set 8	3836	63	99.2	83.3
C4.5 set 9	3836	55	99.3	85.7
C4.5 set 10	3836	16	99.8	88.7

Table A.1: Experiment 1 accuracy (unpruned trees)

	Training Size	Ruleset Size	Training Accuracy%	Test Accuracy%
C4.5 (all)	38360	16	98.7	98.7
C4.5 set 1	3836	4	98.9	98.3
C4.5 set 2	3836	1	98.6	98.6
C4.5 set 3	3836	8	98.7	98.2
C4.5 set 4	3836	5	98.6	98.3
C4.5 set 5	3836	1	98.6	98.6
C4.5 set 6	3836	1	98.4	98.6
C4.5 set 7	3836	1	98.6	98.6
C4.5 set 8	3836	1	98.3	98.6
C4.5 set 9	3836	5	98.5	92.0
C4.5 set 10	3836	1	99.5	98.6

Table A.2: Experiment 1 accuracy (pruned trees)

	Training Size	Ruleset Size	Training Accuracy%	Test Accuracy%
C4.5 (all)	38360	36	98.9	98.9
C4.5 set 1	3836	11	99.3	96.1
C4.5 set 2	3836	17	99.4	88.7
C4.5 set 3	3836	11	99.1	96.3
C4.5 set 4	3836	14	99.1	88.8
C4.5 set 5	3836	18	99.5	77.3
C4.5 set 6	3836	15	99.0	94.7
C4.5 set 7	3836	18	99.4	76.2
C4.5 set 8	3836	12	98.8	81.1
C4.5 set 9	3836	20	99.0	86.4
C4.5 set 10	3836	6	99.7	93.9
FOIL	38360	227	99.2	99.2

Table A.3: Experiment 1 accuracy (rules)

	Training%			Test%		
	Incorrect	None	Multiple	Incorrect	None	Multiple
C4.5 (all)	0.6	0.0	0.0	0.6	0.0	0.0
C4.5 set 1	0.5	0.0	0.0	3.6	0.0	0.0
C4.5 set 2	0.4	0.0	0.0	12.0	0.0	0.0
C4.5 set 3	0.5	0.0	0.0	3.2	0.0	0.0
C4.5 set 4	0.5	0.0	0.0	10.5	0.0	0.0
C4.5 set 5	0.4	0.0	0.0	15.6	0.0	0.0
C4.5 set 6	0.7	0.0	0.0	6.2	0.0	0.0
C4.5 set 7	0.5	0.0	0.0	17.5	0.0	0.0
C4.5 set 8	0.8	0.0	0.0	16.7	0.0	0.0
C4.5 set 9	0.7	0.0	0.0	14.3	0.0	0.0
C4.5 set 10	0.2	0.0	0.0	11.3	0.0	0.0

Table A.4: Experiment 1 errors (unpruned trees)

	Training%			Test%		
	Incorrect	None	Multiple	Incorrect	None	Multiple
C4.5 (all)	1.3	0.0	0.0	1.3	0.0	0.0
C4.5 set 1	1.1	0.0	0.0	1.7	0.0	0.0
C4.5 set 2	1.4	0.0	0.0	1.4	0.0	0.0
C4.5 set 3	1.3	0.0	0.0	1.8	0.0	0.0
C4.5 set 4	1.4	0.0	0.0	1.7	0.0	0.0
C4.5 set 5	1.4	0.0	0.0	1.4	0.0	0.0
C4.5 set 6	1.6	0.0	0.0	1.4	0.0	0.0
C4.5 set 7	1.4	0.0	0.0	1.4	0.0	0.0
C4.5 set 8	1.7	0.0	0.0	1.4	0.0	0.0
C4.5 set 9	1.5	0.0	0.0	8.0	0.0	0.0
C4.5 set 10	0.5	0.0	0.0	1.4	0.0	0.0

Table A.5: Experiment 1 errors (pruned trees)

	Training%			Test%		
	Incorrect	None	Multiple	Incorrect	None	Multiple
C4.5 (all)	1.1	0.0	0.0	1.1	0.0	0.0
C4.5 set 1	0.7	0.0	0.0	3.9	0.0	0.0
C4.5 set 2	0.6	0.0	0.0	11.3	0.0	0.0
C4.5 set 3	0.9	0.0	0.0	3.7	0.0	0.0
C4.5 set 4	0.9	0.0	0.0	11.2	0.0	0.0
C4.5 set 5	0.5	0.0	0.0	22.7	0.0	0.0
C4.5 set 6	1.0	0.0	0.0	5.3	0.0	0.0
C4.5 set 7	0.6	0.0	0.0	23.8	0.0	0.0
C4.5 set 8	1.2	0.0	0.0	18.9	0.0	0.0
C4.5 set 9	1.0	0.0	0.0	13.6	0.0	0.0
C4.5 set 10	0.3	0.0	0.0	6.1	0.0	0.0
FOIL	0.0	0.8	0.0	0.0	0.8	0.0

Table A.6: Experiment 1 errors (rules)

	Training%		Test%	
	$T \Rightarrow F$	$F \Rightarrow T$	$T \Rightarrow F$	$F \Rightarrow T$
C4.5 (all)	38.2	0.1	38.2	0.1
C4.5 set 1	43.5	0.0	90.7	2.4
C4.5 set 2	25.0	0.1	84.3	11.0
C4.5 set 3	25.0	0.1	85.1	2.0
C4.5 set 4	27.4	0.0	77.6	9.5
C4.5 set 5	26.9	0.1	73.1	14.7
C4.5 set 6	40.3	0.1	85.6	5.0
C4.5 set 7	26.9	0.1	79.1	16.6
C4.5 set 8	32.8	0.2	84.1	15.8
C4.5 set 9	27.3	0.2	80.6	13.3
C4.5 set 10	40.0	0.0	84.5	10.2

Table A.7: Experiment 1 misclassifications (unpruned trees)

	Training%		Test%	
	$T \Rightarrow F$	$F \Rightarrow T$	$T \Rightarrow F$	$F \Rightarrow T$
C4.5 (all)	93.7	0.0	93.7	0.0
C4.5 set 1	89.1	0.0	98.9	0.4
C4.5 set 2	100.0	0.0	100.0	0.0
C4.5 set 3	83.3	0.0	92.2	0.5
C4.5 set 4	85.5	0.0	95.7	0.4
C4.5 set 5	100.0	0.0	100.0	0.0
C4.5 set 6	100.0	0.0	100.0	0.0
C4.5 set 7	100.0	0.0	100.0	0.0
C4.5 set 8	100.0	0.0	100.0	0.0
C4.5 set 9	89.4	0.0	76.7	6.7
C4.5 set 10	100.0	0.0	100.0	0.0

Table A.8: Experiment 1 misclassifications (pruned trees)

	Training%		Test%	
	$T \Rightarrow F$	$F \Rightarrow T$	$T \Rightarrow F$	$F \Rightarrow T$
C4.5 (all)	79.9	0.0	79.9	0.0
C4.5 set 1	56.5	0.0	91.6	2.6
C4.5 set 2	36.5	0.1	86.2	10.2
C4.5 set 3	60.0	0.0	87.1	2.5
C4.5 set 4	50.0	0.1	78.4	10.3
C4.5 set 5	34.6	0.1	70.3	22.0
C4.5 set 6	58.1	0.0	93.1	4.1
C4.5 set 7	32.3	0.1	75.4	23.0
C4.5 set 8	61.1	0.1	86.8	18.0
C4.5 set 9	47.0	0.2	84.0	12.6
C4.5 set 10	55.0	0.0	93.7	4.9
FOIL	0.0	0.0	0.0	0.0

Table A.9: Experiment 1 misclassifications (rules)

A.1.2 Experiment 2

Examples: 19180; **Positive:** 349, **Negative:** 18831

Baseline accuracy: 98.2%

	Training Size	Ruleset Size	Training Accuracy%	Test Accuracy%
C4.5 (all)	19180	362	99.4	98.4
C4.5 set 1	1918	57	99.1	94.8
C4.5 set 2	1918	68	98.9	94.8
C4.5 set 3	1918	72	98.9	94.7
C4.5 set 4	1918	57	98.7	94.3
C4.5 set 5	1918	38	99.0	93.9
C4.5 set 6	1918	1	99.7	98.2
C4.5 set 7	1918	1	99.9	98.2
C4.5 set 8	1918	1	100.0	98.2
C4.5 set 9	1918	1	100.0	98.2
C4.5 set 10	1918	1	100.0	98.2

Table A.10: Experiment 2 accuracy (unpruned trees)

	Training Size	Ruleset Size	Training Accuracy%	Test Accuracy%
C4.5 (all)	19180	23	98.4	98.4
C4.5 set 1	1918	5	96.9	98.3
C4.5 set 2	1918	5	96.4	98.1
C4.5 set 3	1918	9	96.5	98.2
C4.5 set 4	1918	1	96.7	98.2
C4.5 set 5	1918	4	97.9	98.0
C4.5 set 6	1918	1	99.7	98.2
C4.5 set 7	1918	1	99.9	98.2
C4.5 set 8	1918	1	100.0	98.2
C4.5 set 9	1918	1	100.0	98.2
C4.5 set 10	1918	1	100.0	98.2

Table A.11: Experiment 2 accuracy (pruned trees)

	Training Size	Ruleset Size	Training Accuracy%	Test Accuracy%
C4.5 (all)	19180	12	98.4	98.4
C4.5 set 1	1918	8	97.2	97.1
C4.5 set 2	1918	6	96.7	97.4
C4.5 set 3	1918	9	96.7	98.1
C4.5 set 4	1918	3	96.7	98.2
C4.5 set 5	1918	6	98.0	97.8
C4.5 set 6	1918	1	99.7	98.2
C4.5 set 7	1918	1	99.9	98.2
C4.5 set 8	1918	1	100.0	98.2
C4.5 set 9	1918	1	100.0	98.2
C4.5 set 10	1918	1	100.0	98.2
FOIL	19180	149	99.0	99.0

Table A.12: Experiment 2 accuracy (rules)

	Training%			Test%		
	Incorrect	None	Multiple	Incorrect	None	Multiple
C4.5 (all)	0.6	0.0	0.0	0.6	0.0	0.0
C4.5 set 1	0.9	0.0	0.0	5.2	0.0	0.0
C4.5 set 2	1.1	0.0	0.0	5.2	0.0	0.0
C4.5 set 3	1.1	0.0	0.0	5.3	0.0	0.0
C4.5 set 4	1.3	0.0	0.0	5.7	0.0	0.0
C4.5 set 5	1.0	0.0	0.0	6.1	0.0	0.0
C4.5 set 6	0.3	0.0	0.0	1.8	0.0	0.0
C4.5 set 7	0.1	0.0	0.0	1.8	0.0	0.0
C4.5 set 8	0.0	0.0	0.0	1.8	0.0	0.0
C4.5 set 9	0.0	0.0	0.0	1.8	0.0	0.0
C4.5 set 10	0.0	0.0	0.0	1.8	0.0	0.0

Table A.13: Experiment 2 errors (unpruned trees)

	Training%			Test%		
	Incorrect	None	Multiple	Incorrect	None	Multiple
C4.5 (all)	1.6	0.0	0.0	1.6	0.0	0.0
C4.5 set 1	3.1	0.0	0.0	1.7	0.0	0.0
C4.5 set 2	3.6	0.0	0.0	1.9	0.0	0.0
C4.5 set 3	3.5	0.0	0.0	1.8	0.0	0.0
C4.5 set 4	3.3	0.0	0.0	1.8	0.0	0.0
C4.5 set 5	2.1	0.0	0.0	2.0	0.0	0.0
C4.5 set 6	0.3	0.0	0.0	1.8	0.0	0.0
C4.5 set 7	0.1	0.0	0.0	1.8	0.0	0.0
C4.5 set 8	0.0	0.0	0.0	1.8	0.0	0.0
C4.5 set 9	0.0	0.0	0.0	1.8	0.0	0.0
C4.5 set 10	0.0	0.0	0.0	1.8	0.0	0.0

Table A.14: Experiment 2 errors (pruned trees)

	Training%			Test%		
	Incorrect	None	Multiple	Incorrect	None	Multiple
C4.5 (all)	1.6	0.0	0.0	1.6	0.0	0.0
C4.5 set 1	2.8	0.0	0.0	2.9	0.0	0.0
C4.5 set 2	3.3	0.0	0.0	2.6	0.0	0.0
C4.5 set 3	3.3	0.0	0.0	1.9	0.0	0.0
C4.5 set 4	3.3	0.0	0.0	1.8	0.0	0.0
C4.5 set 5	2.0	0.0	0.0	2.2	0.0	0.0
C4.5 set 6	0.3	0.0	0.0	1.8	0.0	0.0
C4.5 set 7	0.1	0.0	0.0	1.8	0.0	0.0
C4.5 set 8	0.0	0.0	0.0	1.8	0.0	0.0
C4.5 set 9	0.0	0.0	0.0	1.8	0.0	0.0
C4.5 set 10	0.0	0.0	0.0	1.8	0.0	0.0
FOIL	0.0	1.0	0.0	0.0	1.0	0.0

Table A.15: Experiment 2 errors (rules)

	Training%		Test%	
	$T \Rightarrow F$	$F \Rightarrow T$	$T \Rightarrow F$	$F \Rightarrow T$
C4.5 (all)	20.3	0.2	20.3	0.2
C4.5 set 1	18.8	0.2	74.2	4.0
C4.5 set 2	21.5	0.3	70.2	4.0
C4.5 set 3	16.5	0.4	69.3	4.1
C4.5 set 4	30.2	0.3	78.8	4.3
C4.5 set 5	33.3	0.3	79.7	4.7
C4.5 set 6	100.0	0.0	100.0	0.0
C4.5 set 7	100.0	0.0	100.0	0.0
C4.5 set 8	100.0	0.0	100.0	0.0
C4.5 set 9	—	0.0	100.0	0.0
C4.5 set 10	—	0.0	100.0	0.0

Table A.16: Experiment 2 misclassifications (unpruned trees)

	Training%		Test%	
	$T \Rightarrow F$	$F \Rightarrow T$	$T \Rightarrow F$	$F \Rightarrow T$
C4.5 (all)	86.2	0.0	86.2	0.0
C4.5 set 1	85.5	0.0	90.8	0.1
C4.5 set 2	83.6	0.0	91.9	0.2
C4.5 set 3	78.8	0.1	89.1	0.2
C4.5 set 4	100.0	0.0	100.0	0.0
C4.5 set 5	91.1	0.0	94.0	0.0
C4.5 set 6	100.0	0.0	100.0	0.0
C4.5 set 7	100.0	0.0	100.0	0.0
C4.5 set 8	100.0	0.0	100.0	0.0
C4.5 set 9	—	0.0	100.0	0.0
C4.5 set 10	—	0.0	100.0	0.0

Table A.17: Experiment 2 misclassifications (pruned trees)

	Training%		Test%	
	$T \Rightarrow F$	$F \Rightarrow T$	$T \Rightarrow F$	$F \Rightarrow T$
C4.5 (all)	84.5	0.0	84.5	0.0
C4.5 set 1	76.8	0.1	87.7	1.3
C4.5 set 2	79.7	0.0	85.1	1.1
C4.5 set 3	72.9	0.1	87.4	0.3
C4.5 set 4	100.0	0.0	100.0	0.0
C4.5 set 5	82.2	0.1	92.8	0.5
C4.5 set 6	100.0	0.0	100.0	0.0
C4.5 set 7	100.0	0.0	100.0	0.0
C4.5 set 8	100.0	0.0	100.0	0.0
C4.5 set 9	—	0.0	100.0	0.0
C4.5 set 10	—	0.0	100.0	0.0
FOIL	0.0	0.0	0.0	0.0

Table A.18: Experiment 2 misclassifications (rules)

A.1.3 Experiment 3

Examples: 6300; **Positive:** 115, **Negative:** 6185

Baseline accuracy: 98.2%

	Training Size	Ruleset Size	Training Accuracy%	Test Accuracy%
C4.5 (all)	6300	96	99.3	99.3
C4.5 set 1	630	25	99.5	95.3
C4.5 set 2	630	23	99.5	95.5
C4.5 set 3	630	31	99.0	94.9
C4.5 set 4	630	14	98.9	97.4
C4.5 set 5	630	11	99.7	97.8
C4.5 set 6	630	1	100.0	98.2
C4.5 set 7	630	1	100.0	98.2
C4.5 set 8	630	1	100.0	98.2
C4.5 set 9	630	1	100.0	98.2
C4.5 set 10	630	1	100.0	98.2

Table A.19: Experiment 3 accuracy (unpruned trees)

	Training Size	Ruleset Size	Training Accuracy%	Test Accuracy%
C4.5 (all)	6300	6	98.4	98.4
C4.5 set 1	630	12	97.9	97.1
C4.5 set 2	630	7	97.8	97.5
C4.5 set 3	630	4	95.2	98.2
C4.5 set 4	630	1	97.1	98.2
C4.5 set 5	630	1	98.7	98.2
C4.5 set 6	630	1	100.0	98.2
C4.5 set 7	630	1	100.0	98.2
C4.5 set 8	630	1	100.0	98.2
C4.5 set 9	630	1	100.0	98.2
C4.5 set 10	630	1	100.0	98.2

Table A.20: Experiment 3 accuracy (pruned trees)

	Training Size	Ruleset Size	Training Accuracy%	Test Accuracy%
C4.5 (all)	6300	9	98.6	98.6
C4.5 set 1	630	9	98.1	96.3
C4.5 set 2	630	7	98.1	97.5
C4.5 set 3	630	6	96.2	97.7
C4.5 set 4	630	4	97.8	98.1
C4.5 set 5	630	4	99.7	97.8
C4.5 set 6	630	1	100.0	98.2
C4.5 set 7	630	1	100.0	98.2
C4.5 set 8	630	1	100.0	98.2
C4.5 set 9	630	1	100.0	98.2
C4.5 set 10	630	1	100.0	98.2
FOIL	6300	55	99.0	99.0

Table A.21: Experiment 3 accuracy (rules)

	Training%			Test%		
	Incorrect	None	Multiple	Incorrect	None	Multiple
C4.5 (all)	0.7	0.0	0.0	0.7	0.0	0.0
C4.5 set 1	0.5	0.0	0.0	4.7	0.0	0.0
C4.5 set 2	0.5	0.0	0.0	4.5	0.0	0.0
C4.5 set 3	1.0	0.0	0.0	5.1	0.0	0.0
C4.5 set 4	1.1	0.0	0.0	2.6	0.0	0.0
C4.5 set 5	0.3	0.0	0.0	2.2	0.0	0.0
C4.5 set 6	0.0	0.0	0.0	1.8	0.0	0.0
C4.5 set 7	0.0	0.0	0.0	1.8	0.0	0.0
C4.5 set 8	0.0	0.0	0.0	1.8	0.0	0.0
C4.5 set 9	0.0	0.0	0.0	1.8	0.0	0.0
C4.5 set 10	0.0	0.0	0.0	1.8	0.0	0.0

Table A.22: Experiment 3 errors (unpruned trees)

	Training%			Test%		
	Incorrect	None	Multiple	Incorrect	None	Multiple
C4.5 (all)	1.6	0.0	0.0	1.6	0.0	0.0
C4.5 set 1	2.1	0.0	0.0	2.9	0.0	0.0
C4.5 set 2	2.2	0.0	0.0	2.5	0.0	0.0
C4.5 set 3	4.8	0.0	0.0	1.8	0.0	0.0
C4.5 set 4	2.9	0.0	0.0	1.8	0.0	0.0
C4.5 set 5	1.3	0.0	0.0	1.8	0.0	0.0
C4.5 set 6	0.0	0.0	0.0	1.8	0.0	0.0
C4.5 set 7	0.0	0.0	0.0	1.8	0.0	0.0
C4.5 set 8	0.0	0.0	0.0	1.8	0.0	0.0
C4.5 set 9	0.0	0.0	0.0	1.8	0.0	0.0
C4.5 set 10	0.0	0.0	0.0	1.8	0.0	0.0

Table A.23: Experiment 3 errors (pruned trees)

	Training%			Test%		
	Incorrect	None	Multiple	Incorrect	None	Multiple
C4.5 (all)	1.4	0.0	0.0	1.4	0.0	0.0
C4.5 set 1	1.9	0.0	0.0	3.7	0.0	0.0
C4.5 set 2	2.5	0.0	0.0	1.9	0.0	0.0
C4.5 set 3	3.8	0.0	0.0	2.3	0.0	0.0
C4.5 set 4	2.2	0.0	0.0	1.9	0.0	0.0
C4.5 set 5	0.3	0.0	0.0	2.2	0.0	0.0
C4.5 set 6	0.0	0.0	0.0	1.8	0.0	0.0
C4.5 set 7	0.0	0.0	0.0	1.8	0.0	0.0
C4.5 set 8	0.0	0.0	0.0	1.8	0.0	0.0
C4.5 set 9	0.0	0.0	0.0	1.8	0.0	0.0
C4.5 set 10	0.0	0.0	0.0	1.8	0.0	0.0
FOIL	0.0	1.0	0.0	0.0	1.0	0.0

Table A.24: Experiment 3 errors (rules)

	Training%		Test%	
	$T \Rightarrow F$	$F \Rightarrow T$	$T \Rightarrow F$	$F \Rightarrow T$
C4.5 (all)	33.9	0.1	33.9	0.1
C4.5 set 1	6.9	0.2	67.8	3.5
C4.5 set 2	8.3	0.2	71.3	3.2
C4.5 set 3	11.1	0.3	61.7	4.1
C4.5 set 4	27.8	0.3	78.3	1.1
C4.5 set 5	12.5	0.2	92.2	0.5
C4.5 set 6	—	0.0	100.0	0.0
C4.5 set 7	—	0.0	100.0	0.0
C4.5 set 8	—	0.0	100.0	0.0
C4.5 set 9	—	0.0	100.0	0.0
C4.5 set 10	—	0.0	100.0	0.0

Table A.25: Experiment 3 misclassifications (unpruned trees)

	Training%		Test%	
	$T \Rightarrow F$	$F \Rightarrow T$	$T \Rightarrow F$	$F \Rightarrow T$
C4.5 (all)	87.0	0.0	87.0	0.0
C4.5 set 1	44.8	0.0	79.1	1.4
C4.5 set 2	58.3	0.0	84.3	1.0
C4.5 set 3	83.3	0.0	92.2	0.1
C4.5 set 4	100.0	0.0	100.0	0.0
C4.5 set 5	100.0	0.0	100.0	0.0
C4.5 set 6	—	0.0	100.0	0.0
C4.5 set 7	—	0.0	100.0	0.0
C4.5 set 8	—	0.0	100.0	0.0
C4.5 set 9	—	0.0	100.0	0.0
C4.5 set 10	—	0.0	100.0	0.0

Table A.26: Experiment 3 misclassifications (pruned trees)

	Training%		Test%	
	$T \Rightarrow F$	$F \Rightarrow T$	$T \Rightarrow F$	$F \Rightarrow T$
C4.5 (all)	75.7	0.0	75.7	0.0
C4.5 set 1	41.4	0.0	76.5	2.4
C4.5 set 2	50.0	0.0	80.9	1.1
C4.5 set 3	63.9	0.2	80.9	0.9
C4.5 set 4	66.7	0.3	87.8	0.3
C4.5 set 5	12.5	0.2	92.2	0.5
C4.5 set 6	—	0.0	100.0	0.0
C4.5 set 7	—	0.0	100.0	0.0
C4.5 set 8	—	0.0	100.0	0.0
C4.5 set 9	—	0.0	100.0	0.0
C4.5 set 10	—	0.0	100.0	0.0
FOIL	0.0	0.0	0.0	0.0

Table A.27: Experiment 3 misclassifications (rules)

A.1.4 Experiment 4

Examples: 14037; **Positive:** 368, **Negative:** 13669

Baseline accuracy: 97.4%

	Training Size	Ruleset Size	Training Accuracy%	Test Accuracy%
C4.5 1%	512	64	93.4	40.8
C4.5 2%	638	94	94.5	49.6
C4.5 5%	1036	152	95.2	66.4
C4.5 10%	1718	193	92.9	80.6
C4.5 20%	3080	249	93.2	86.9
C4.5 50%	7162	286	92.7	91.3
C4.5 100%	14037	340	94.0	94.0

Table A.28: Experiment 4 accuracy (unpruned trees)

	Training Size	Ruleset Size	Training Accuracy%	Test Accuracy%
C4.5 1%	512	60	93.2	40.5
C4.5 2%	638	87	93.9	49.7
C4.5 5%	1036	139	95.0	66.1
C4.5 10%	1718	157	91.6	80.6
C4.5 20%	3080	47	87.9	92.0
C4.5 50%	7162	21	94.5	96.3
C4.5 100%	14037	11	96.7	96.7

Table A.29: Experiment 4 accuracy (pruned trees)

	Training Size	Ruleset Size	Training Accuracy%	Test Accuracy%
C4.5 1%	512	7	73.8	5.9
C4.5 2%	638	9	68.0	44.1
C4.5 5%	1036	12	76.1	92.2
C4.5 10%	1718	12	84.7	95.6
C4.5 20%	3080	7	90.3	96.9
C4.5 50%	7162	8	95.5	97.5
C4.5 100%	14037	12	97.7	97.7
FOIL 1%	512	40	96.9	18.5
FOIL 2%	638	60	98.6	29.2
FOIL 5%	1036	77	96.6	53.2
FOIL 10%	1718	99	96.9	73.2
FOIL 20%	3080	118	97.2	84.7
FOIL 50%	7162	131	96.6	94.7
FOIL 100%	14037	129	97.3	97.3

Table A.30: Experiment 4 accuracy (rules)

	Training%			Test%		
	Incorrect	None	Multiple	Incorrect	None	Multiple
C4.5 1%	3.5	3.1	0.0	55.3	3.8	0.0
C4.5 2%	2.4	3.1	0.0	47.0	3.4	0.0
C4.5 5%	2.7	2.1	0.0	30.6	3.1	0.0
C4.5 10%	2.6	4.5	0.0	15.6	4.5	0.0
C4.5 20%	1.9	4.9	0.0	8.3	4.8	0.0
C4.5 50%	1.8	5.6	0.0	3.1	5.5	0.0
C4.5 100%	1.3	4.7	0.0	1.3	4.7	0.0

Table A.31: Experiment 4 errors (unpruned trees)

	Training%			Test%		
	Incorrect	None	Multiple	Incorrect	None	Multiple
C4.5 1%	3.7	3.1	0.0	55.8	3.7	0.0
C4.5 2%	2.7	3.4	0.0	47.1	3.2	0.0
C4.5 5%	2.9	2.1	0.0	30.9	3.0	0.0
C4.5 10%	3.9	4.4	0.0	15.0	4.4	0.0
C4.5 20%	7.9	4.2	0.0	3.9	4.1	0.0
C4.5 50%	4.3	1.2	0.0	2.4	1.3	0.0
C4.5 100%	2.3	1.0	0.0	2.3	1.0	0.0

Table A.32: Experiment 4 errors (pruned trees)

	Training%			Test%		
	Incorrect	None	Multiple	Incorrect	None	Multiple
C4.5 1%	26.2	0.0	0.0	94.1	0.0	0.0
C4.5 2%	32.0	0.0	0.0	55.9	0.0	0.0
C4.5 5%	23.9	0.0	0.0	7.8	0.0	0.0
C4.5 10%	15.3	0.0	0.0	4.4	0.0	0.0
C4.5 20%	9.7	0.0	0.0	3.1	0.0	0.0
C4.5 50%	4.5	0.0	0.0	2.5	0.0	0.0
C4.5 100%	2.3	0.0	0.0	2.3	0.0	0.0
FOIL 1%	0.2	2.9	0.0	42.5	21.4	17.6
FOIL 2%	0.0	1.4	0.0	27.5	21.2	22.1
FOIL 5%	0.1	3.3	0.0	11.3	18.2	17.3
FOIL 10%	0.1	2.9	0.0	4.7	11.8	10.3
FOIL 20%	0.2	2.5	0.0	1.9	7.3	6.1
FOIL 50%	0.5	3.2	0.0	0.5	3.8	1.0
FOIL 100%	0.1	2.7	1.0	0.0	2.7	0.0

Table A.33: Experiment 4 errors (rules)

	Training%		Test%	
	$T \Rightarrow F$	$F \Rightarrow T$	$T \Rightarrow F$	$F \Rightarrow T$
C4.5 1%	3.1	5.0	3.1	59.0
C4.5 2%	1.7	3.4	1.7	49.9
C4.5 5%	4.7	1.7	4.7	32.3
C4.5 10%	7.3	1.6	7.3	16.6
C4.5 20%	10.3	0.9	10.3	8.7
C4.5 50%	27.0	0.5	27.0	2.7
C4.5 100%	37.2	0.4	37.2	0.4

Table A.34: Experiment 4 misclassifications (unpruned trees)

	Training%		Test%	
	$T \Rightarrow F$	$F \Rightarrow T$	$T \Rightarrow F$	$F \Rightarrow T$
C4.5 1%	3.1	5.7	3.1	59.4
C4.5 2%	1.7	4.2	1.7	49.9
C4.5 5%	3.6	2.6	3.6	32.6
C4.5 10%	11.8	2.1	11.8	15.8
C4.5 20%	67.0	0.3	67.0	2.3
C4.5 50%	84.1	0.0	84.1	0.3
C4.5 100%	88.9	0.1	88.9	0.1

Table A.35: Experiment 4 misclassifications (pruned trees)

	Training%		Test%	
	$T \Rightarrow F$	$F \Rightarrow T$	$T \Rightarrow F$	$F \Rightarrow T$
C4.5 1%	2.2	93.1	0.0	96.6
C4.5 2%	23.9	43.0	23.9	56.8
C4.5 5%	67.4	0.0	67.4	6.2
C4.5 10%	73.9	0.1	73.9	2.5
C4.5 20%	81.3	0.0	81.3	1.0
C4.5 50%	87.2	0.0	87.2	0.2
C4.5 100%	87.0	0.1	87.0	0.1
FOIL 1%	0.0	0.8	0.0	64.2
FOIL 2%	0.0	0.0	0.0	50.4
FOIL 5%	0.3	0.0	0.0	29.6
FOIL 10%	0.0	0.2	0.0	15.6
FOIL 20%	0.3	0.3	0.3	8.3
FOIL 50%	2.1	0.1	2.1	1.6
FOIL 100%	2.7	0.1	2.7	0.1

Table A.36: Experiment 4 misclassifications (rules)

A.1.5 Experiment 5

Examples: 14548; **Positive:** 371, **Negative:** 14177

Baseline accuracy: 97.4%

	Training Size	Ruleset Size	Training Accuracy%	Test Accuracy%
C4.5 1%	520	61	91.5	39.7
C4.5 2%	656	87	91.6	54.6
C4.5 5%	1084	127	93.0	70.7
C4.5 10%	1796	190	94.4	82.3
C4.5 20%	3203	224	94.4	88.9
C4.5 50%	7481	284	95.0	93.2
C4.5 100%	14548	325	95.2	95.2

Table A.37: Experiment 5 accuracy (unpruned trees)

	Training Size	Ruleset Size	Training Accuracy%	Test Accuracy%
C4.5 1%	520	52	91.2	40.8
C4.5 2%	656	78	91.2	53.8
C4.5 5%	1084	116	93.0	70.2
C4.5 10%	1796	170	94.4	81.4
C4.5 20%	3203	98	90.9	91.4
C4.5 50%	7481	49	93.3	94.3
C4.5 100%	14548	25	96.4	96.4

Table A.38: Experiment 5 accuracy (pruned trees)

	Training Size	Ruleset Size	Training Accuracy%	Test Accuracy%
C4.5 1%	520	8	62.7	77.7
C4.5 2%	656	7	72.1	66.8
C4.5 5%	1084	11	80.8	90.1
C4.5 10%	1796	11	84.9	95.8
C4.5 20%	3203	18	91.8	96.7
C4.5 50%	7481	11	95.9	97.7
C4.5 100%	14548	15	98.0	98.0
FOIL 1%	520	35	99.2	18.8
FOIL 2%	656	51	98.5	35.9
FOIL 5%	1084	77	99.2	53.8
FOIL 10%	1796	95	99.7	74.2
FOIL 20%	3203	112	99.1	86.3
FOIL 50%	7481	124	98.7	95.9
FOIL 100%	14548	127	98.9	98.9

Table A.39: Experiment 5 accuracy (rules)

	Training%			Test%		
	Incorrect	None	Multiple	Incorrect	None	Multiple
C4.5 1%	3.3	5.2	0.0	56.6	3.6	0.0
C4.5 2%	3.5	4.9	0.0	41.8	3.6	0.0
C4.5 5%	2.7	4.3	0.0	25.6	3.6	0.0
C4.5 10%	1.6	4.0	0.0	13.8	3.9	0.0
C4.5 20%	1.3	4.2	0.0	6.8	4.2	0.0
C4.5 50%	0.9	4.1	0.0	2.6	4.2	0.0
C4.5 100%	0.6	4.2	0.0	0.6	4.2	0.0

Table A.40: Experiment 5 errors (unpruned trees)

	Training%			Test%		
	Incorrect	None	Multiple	Incorrect	None	Multiple
C4.5 1%	3.7	5.2	0.0	55.6	3.6	0.0
C4.5 2%	4.0	4.9	0.0	42.6	3.6	0.0
C4.5 5%	2.7	4.3	0.0	26.1	3.6	0.0
C4.5 10%	1.8	3.7	0.0	14.9	3.7	0.0
C4.5 20%	4.8	4.2	0.0	4.4	4.2	0.0
C4.5 50%	3.4	3.4	0.0	2.2	3.5	0.0
C4.5 100%	2.0	1.6	0.0	2.0	1.6	0.0

Table A.41: Experiment 5 errors (pruned trees)

	Training%			Test%		
	Incorrect	None	Multiple	Incorrect	None	Multiple
C4.5 1%	37.3	0.0	0.0	22.3	0.0	0.0
C4.5 2%	27.9	0.0	0.0	33.2	0.0	0.0
C4.5 5%	19.2	0.0	0.0	9.9	0.0	0.0
C4.5 10%	15.1	0.0	0.0	4.2	0.0	0.0
C4.5 20%	8.2	0.0	0.0	3.3	0.0	0.0
C4.5 50%	4.1	0.0	0.0	2.3	0.0	0.0
C4.5 100%	2.0	0.0	0.0	2.0	0.0	0.0
FOIL 1%	0.0	0.8	0.0	41.3	20.8	19.1
FOIL 2%	0.0	1.5	0.0	23.0	19.2	21.9
FOIL 5%	0.0	0.7	0.1	9.2	16.4	20.6
FOIL 10%	0.0	0.2	0.1	3.6	10.3	11.9
FOIL 20%	0.0	0.8	0.1	1.2	6.1	6.4
FOIL 50%	0.0	1.2	0.1	0.2	2.5	1.4
FOIL 100%	0.0	1.1	0.0	0.0	1.1	0.0

Table A.42: Experiment 5 errors (rules)

	Training%		Test%	
	$T \Rightarrow F$	$F \Rightarrow T$	$T \Rightarrow F$	$F \Rightarrow T$
C4.5 1%	2.5	5.8	2.5	60.2
C4.5 2%	3.4	4.1	3.4	44.4
C4.5 5%	4.2	2.0	4.2	27.2
C4.5 10%	4.3	1.0	4.3	14.7
C4.5 20%	7.1	0.6	7.1	7.1
C4.5 50%	11.0	0.4	11.0	2.5
C4.5 100%	18.4	0.2	18.4	0.2

Table A.43: Experiment 5 misclassifications (unpruned trees)

	Training%		Test%	
	$T \Rightarrow F$	$F \Rightarrow T$	$T \Rightarrow F$	$F \Rightarrow T$
C4.5 1%	3.1	5.8	3.1	59.1
C4.5 2%	2.8	5.9	2.8	45.3
C4.5 5%	4.0	2.2	4.0	27.7
C4.5 10%	4.3	1.3	4.3	15.7
C4.5 20%	41.0	0.3	41.0	3.6
C4.5 50%	70.5	0.1	70.5	0.5
C4.5 100%	80.8	0.0	80.8	0.0

Table A.44: Experiment 5 misclassifications (pruned trees)

	Training%		Test%	
	$T \Rightarrow F$	$F \Rightarrow T$	$T \Rightarrow F$	$F \Rightarrow T$
C4.5 1%	50.1	5.4	50.1	21.5
C4.5 2%	30.7	24.2	30.7	33.3
C4.5 5%	55.8	0.1	55.8	8.7
C4.5 10%	73.3	0.0	73.3	2.4
C4.5 20%	68.6	0.0	68.6	1.6
C4.5 50%	83.6	0.0	83.6	0.1
C4.5 100%	78.7	0.0	78.7	0.0
FOIL 1%	0.0	0.0	0.0	63.1
FOIL 2%	0.0	0.0	0.0	44.8
FOIL 5%	0.3	0.0	0.3	29.3
FOIL 10%	0.0	0.1	0.0	15.6
FOIL 20%	0.0	0.1	0.0	7.8
FOIL 50%	0.0	0.1	0.0	1.6
FOIL 100%	0.0	0.0	0.0	0.0

Table A.45: Experiment 5 misclassifications (rules)

A.1.6 Experiment 5a

Examples: 14548; **Positive:** 371, **Negative:** 14177

Baseline accuracy: 97.4%

	Training Size	Ruleset Size	Training Accuracy%	Test Accuracy%
C4.5 1%	520	63	72.1	41.0
C4.5 2%	656	80	75.2	55.6
C4.5 5%	1084	125	81.5	68.8
C4.5 10%	1796	159	82.7	75.9
C4.5 20%	3203	178	84.0	81.5
C4.5 50%	7481	209	87.0	85.7
C4.5 100%	14548	277	86.3	86.3

Table A.46: Experiment 5a accuracy (unpruned trees)

	Training Size	Ruleset Size	Training Accuracy%	Test Accuracy%
C4.5 1%	520	32	70.6	43.0
C4.5 2%	656	61	75.8	61.1
C4.5 5%	1084	75	81.5	76.2
C4.5 10%	1796	78	82.6	81.4
C4.5 20%	3203	88	84.8	85.0
C4.5 50%	7481	79	88.0	88.1
C4.5 100%	14548	61	88.5	88.5

Table A.47: Experiment 5a accuracy (pruned trees)

	Training Size	Ruleset Size	Training Accuracy%	Test Accuracy%
C4.5 1%	520	6	89.2	61.8
C4.5 2%	656	8	85.1	59.8
C4.5 5%	1084	6	79.5	89.2
C4.5 10%	1796	13	86.9	95.3
C4.5 20%	3203	11	89.3	91.9
C4.5 50%	7481	13	96.2	97.7
C4.5 100%	14548	15	97.8	97.8
FOIL 1%	520	33	98.7	28.6
FOIL 2%	656	47	98.9	46.7
FOIL 5%	1084	64	99.1	65.7
FOIL 10%	1796	76	99.2	79.0
FOIL 20%	3203	97	99.4	87.0
FOIL 50%	7481	111	99.1	95.8
FOIL 100%	14548	112	99.0	99.0

Table A.48: Experiment 5a accuracy (rules)

	Training%			Test%		
	Incorrect	None	Multiple	Incorrect	None	Multiple
C4.5 1%	2.3	25.6	0.0	46.9	12.1	0.0
C4.5 2%	1.8	23.0	0.0	32.3	12.1	0.0
C4.5 5%	1.4	17.1	0.0	19.0	12.2	0.0
C4.5 10%	1.7	15.5	0.0	12.0	12.1	0.0
C4.5 20%	1.3	14.7	0.0	5.8	12.7	0.0
C4.5 50%	0.6	12.4	0.0	1.9	12.4	0.0
C4.5 100%	0.5	13.2	0.0	0.5	13.2	0.0

Table A.49: Experiment 5a errors (unpruned trees)

	Training%			Test%		
	Incorrect	None	Multiple	Incorrect	None	Multiple
C4.5 1%	4.2	25.2	0.0	45.5	11.5	0.0
C4.5 2%	1.5	22.7	0.0	27.3	11.5	0.0
C4.5 5%	1.8	16.7	0.0	12.7	11.1	0.0
C4.5 10%	2.4	15.0	0.0	7.4	11.3	0.0
C4.5 20%	1.7	13.5	0.0	3.7	11.2	0.0
C4.5 50%	1.2	10.7	0.0	1.3	10.6	0.0
C4.5 100%	1.0	10.5	0.0	1.0	10.5	0.0

Table A.50: Experiment 5a errors (pruned trees)

	Training%			Test%		
	Incorrect	None	Multiple	Incorrect	None	Multiple
C4.5 1%	10.8	0.0	0.0	38.2	0.0	0.0
C4.5 2%	14.9	0.0	0.0	40.2	0.0	0.0
C4.5 5%	20.5	0.0	0.0	10.8	0.0	0.0
C4.5 10%	13.1	0.0	0.0	4.7	0.0	0.0
C4.5 20%	10.7	0.0	0.0	8.1	0.0	0.0
C4.5 50%	3.8	0.0	0.0	2.3	0.0	0.0
C4.5 100%	2.2	0.0	0.0	2.2	0.0	0.0
FOIL 1%	0.0	1.3	0.0	26.6	12.6	32.2
FOIL 2%	0.0	1.1	0.0	14.6	13.2	25.4
FOIL 5%	0.0	0.8	0.1	7.9	11.0	15.4
FOIL 10%	0.0	0.5	0.3	2.9	7.8	10.3
FOIL 20%	0.0	0.5	0.1	1.5	5.4	6.1
FOIL 50%	0.0	0.8	0.1	0.2	2.3	1.6
FOIL 100%	0.0	0.9	0.0	0.0	0.9	0.0

Table A.51: Experiment 5a errors (rules)

	Training%		Test%	
	$T \Rightarrow F$	$F \Rightarrow T$	$T \Rightarrow F$	$F \Rightarrow T$
C4.5 1%	1.5	6.3	1.5	54.4
C4.5 2%	1.2	3.6	1.2	37.5
C4.5 5%	1.1	1.9	1.1	22.1
C4.5 10%	4.5	1.5	4.5	13.8
C4.5 20%	6.8	0.9	6.8	6.6
C4.5 50%	6.5	0.4	6.5	2.0
C4.5 100%	15.4	0.3	15.4	0.3

Table A.52: Experiment 5a misclassifications (unpruned trees)

	Training%		Test%	
	$T \Rightarrow F$	$F \Rightarrow T$	$T \Rightarrow F$	$F \Rightarrow T$
C4.5 1%	1.1	14.8	1.1	52.5
C4.5 2%	1.9	2.0	1.9	31.5
C4.5 5%	5.3	0.8	5.3	14.5
C4.5 10%	10.9	1.1	10.9	8.2
C4.5 20%	16.3	0.4	16.3	3.9
C4.5 50%	31.7	0.1	31.7	0.8
C4.5 100%	51.5	0.0	51.5	0.0

Table A.53: Experiment 5a misclassifications (pruned trees)

	Training%		Test%	
	$T \Rightarrow F$	$F \Rightarrow T$	$T \Rightarrow F$	$F \Rightarrow T$
C4.5 1%	4.0	27.5	4.0	39.1
C4.5 2%	2.2	31.6	2.2	41.2
C4.5 5%	44.7	7.6	44.7	9.9
C4.5 10%	61.7	0.8	61.7	3.2
C4.5 20%	42.9	6.5	42.9	7.2
C4.5 50%	76.5	0.0	76.5	0.3
C4.5 100%	80.6	0.1	80.6	0.1
FOIL 1%	0.0	0.0	0.0	50.2
FOIL 2%	0.0	0.0	0.0	36.5
FOIL 5%	0.0	0.1	0.0	22.9
FOIL 10%	0.0	0.3	0.0	13.2
FOIL 20%	0.0	0.1	0.0	7.7
FOIL 50%	0.0	0.1	0.0	1.9
FOIL 100%	0.0	0.0	0.0	0.0

Table A.54: Experiment 5a misclassifications (rules)

A.2 Sequence Identification Experiments

The output produced by the HTK model testing program produces a confusion matrix and overall percentage recognition accuracy for each testing run. The tables below present this information for all of the sequence identification experiments described in Chapter 3. These tables also indicate the sizes of the training and test sets used, and the baseline accuracy of the dataset.

A.2.1 Experiment 6

Sequences: 703; **Positive:** 359, **Negative:** 344

Training size: 466; **Test size:** 237; **Baseline accuracy:** 51.1%

Prototype	Test % Accuracy	Test % $T \Rightarrow F$	Test % $F \Rightarrow T$
1	63.29	18.6	54.6
2	63.29	22.0	51.3
3	58.65	39.0	43.7
4	64.98	19.5	50.4

Table A.55: Experiment 6 test accuracy

A.2.2 Experiment 7

Sequences: 810; **Positive:** 385, **Negative:** 425

Training size: 574; **Test size:** 263; **Baseline accuracy:** 52.5%

Filter	Test % Accuracy	Test % $T \Rightarrow F$	Test % $F \Rightarrow T$
Order 2 LP	55.02	46.0	44.0
Order 3 LP	57.03	45.7	40.4
Order 5 LP	53.23	45.7	47.8
Order 8 LP	53.61	48.8	44.1
Order 10 LP	59.32	53.5	28.7
Order 12 LP	54.37	44.9	46.3
Order 10 LP Cepstra	61.60	54.3	23.5
Order 12 LP Cepstra	58.17	62.8	31.6

Table A.56: Experiment 7 test accuracy

Appendix B

Hidden Markov Model Prototypes

This appendix presents the initial state transition matrices used in Experiments 6 and 7. These *prototype* models form the basis for the models learnt by the HTK re-estimation tools. Row i of a transition matrix represents the probabilities for transitions out of state i of the model. The first and last states (rows) of an HTK model are non-emitting states, that is they do not generate an output symbol. Essentially they are placeholder states—the first state defines the probability of the first output being generated by each of the other states and the last state is an "exit" state which is entered when the model is ready to terminate. These two states can safely be ignored when designing a prototype model.

Transitions that are marked with zeroes in the prototype matrices are not changed by any of the HTK tools, thus will always remain set to zero. These elements indicate transitions that are not allowed. The non-zero elements of each row give a weighting to the permitted transitions. The entries in each row need not add to one—they will be scaled appropriately by the HTK system.

B.1 Experiment 6

The sequences used in Experiment 6 consisted of 14 milkings (7 days) of a particular cow. All of the prototype models presented below are examples of *left-right* models, since the only transitions allowed are those to the current or later states. Transitions back to a previous state are not permitted. The first prototype uses seven internal states, one for each day represented in the sequence. The second and third models extend this idea by having a state for each milking in the sequence—14 internal states in total. The fourth model used in this experiment has only two internal states, that do not have any intentional correspondence with the samples making up each sequence.

$$\begin{bmatrix} 0 & 1.0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.5 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & 0.5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.5 & 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0.5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.5 & 0.5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.5 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Figure B.1: Experiment 6 prototype model 1

0	1.0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1.0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1.0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1.0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1.0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1.0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1.0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1.0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1.0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1.0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1.0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1.0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1.0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure B.2: Experiment 6 prototype model 2

0.2	0.7	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0.2	0.7	0.1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0.2	0.7	0.1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0.2	0.7	0.1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0.2	0.7	0.1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0.2	0.7	0.1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0.2	0.7	0.1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0.2	0.7	0.1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0.2	0.7	0.1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0.2	0.7	0.1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0.2	0.7	0.1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0.2	0.7	0.1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0.2	0.7	0.1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0.2	0.7	0.1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure B.3: Experiment 6 prototype model 3

$$\begin{bmatrix} 0.1 & 0.7 & 0.2 & 0 \\ 0 & 0.1 & 0.7 & 0.2 \\ 0 & 0 & 0.1 & 0.9 \\ 0 & 0 & 0 & 1.0 \end{bmatrix}$$

Figure B.4: Experiment 6 prototype model 4

B.2 Experiment 7

This experiment made use of sequences of values from a single days' milkings. There is no obvious relationship between samples in these sequences and states in a model. Both models presented below use three internal states. The first model is a simple left-right design, similar to those used in Experiment 6 above. This model was the prototype for the models trained from standard LP encoded sequences. The second model was used with the sequences encoded using LP cepstral and cepstral delta coefficients. All of the entries in this transition matrix are equal, meaning that any state transition is possible. This design allows HTK to define a model structure entirely based on the training sequences—transitions which are unimportant will become zero and the probabilities on the best path through the model will be increased.

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0.5 & 0.5 & 0 & 0 \\ 0 & 0 & 0.5 & 0.5 & 0 \\ 0 & 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

LP encoded models

$$\begin{bmatrix} 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\ 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\ 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\ 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\ 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \end{bmatrix}$$

LP Cepstral encoded models

Figure B.5: Experiment 7 prototype models