MACHINE LEARNING IN ARTIFICIAL INTELLIGENCE

A THESIS SUBMITTED TO THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES OF ÇANKAYA UNIVERSITY

BY

TARDU ERCAN

IN PARTIAL FULLFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE IN COMPUTER ENGINEERING

May 2006

Title of the Thesis : **Machine Learning in Artificial Intelligence**

Submitted by **Tardu ERCAN**

Approval of the Graduate School of Natural and Applied Sciences, Çankaya University

[illegible]

Prof. Dr. Yurdahan GÜLER

Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

[illegible]

Prof. Dr. Mehmet R.TOLUN

Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

[illegible]

Dr. Ali Rıza AŞKUN

Supervisor

Examination Date : 17.05.2006

Examining Committee Members

Prof. Dr. Mehmet Reşit TOLUN	(Çankaya Univ.)	[illegible]
Dr. Ali Rıza AŞKUN	(Çankaya Univ.)	[illegible]
Assoc. Prof. Dr. Ferda Nur ALPASLAN	(METU)	[illegible]

STATEMENT OF NON-PLAGIARISM PAGE

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name	: Tardu ERCAN
Signature	: [illegible]
Date	: 17.05.2006

ABSTRACT

PERFORMANCE ANALYSIS OF DECISION TREE ALGORITHMS ON WATER-CONSUMPTION DOMAIN

ERCAN, Tardu

M.S.c., Department of Computer Engineering

Supervisor: Dr. Ali Rıza AŞKUN

May 2006, 76 Pages

In today's world, learning is a process of computers as well as human being. "Learnable" systems and computers will become more important in following years and affect our lives in many ways. In this thesis, a survey has been carried out in the field of artificial intelligence, machine learning and especially on decision tree learning algorithms. Some of the decision tree learning algorithms was used to learn rules which are extracted from a dataset. The dataset which consists of water consumption of Ankara for one year and meteorological data of Ankara was used. The results indicate that which learning method is more efficient and have better performance.

Keywords: Artificial Intelligence, Machine Learning, Decision Tree Algorithms

ÖZ

KARAR AĞACI ALGORİTMALARININ SU TÜKETİM ALANI ÜSTÜNDE UYGULANAN PERFORMANS ANALİZİ

ERCAN, Tardu

Yüksek Lisans, Bilgisayar Mühendisliği Anabilim Dalı

Tez Yöneticisi : Dr. Ali Rıza AŞKUN

Mayıs 2006, 76 sayfa

Bugünün dünyasında, “Öğrenme” insanların yaptığı kadar, bilgisayarlarında yaptığı bir eylem haline gelmiştir. “Öğrenebilen” sistemler ve bilgisayarlar önümüzdeki yıllarda, hayatımızda çok daha fazla yer alacaklar.Bu tez içinde, yapay zeka ve öğrenme, özellikle karar verme ağacı algoritmalarının yapısı üzerinde çalışılmıştır Öğrenme kurallarını, veri setimizden çıkarmak için bazı algoritmalar kullanılmıştır. Bu veri setimizde Ankara`nın bir yıllık su tüketim oranı ve bir yıllık meteoroloji bilgileri vardır.

Ortaya çıkan sonuçlar ışığında hangi algoritmanın daha verimli ve daha iyi performansa sahip olduğuna işaret eder.

Anahtar Sözcükler: Yapay Zeka, Öğrenme,Karar Ağacı Algoritması.

ACKNOWLEDGEMENTS

I would like to declare my sincere gratefulness to my supervisor Ali Rıza AŞKUN for his enormous support and guidance during this thesis.

I would like to thanks Prof. Dr Mehmet Reşit TOLUN and Assoc. Prof. Ferda Nur ALPASLAN for reading and commenting on this thesis.

I would also like to thank to a special friend, Zeki YETGİN, who showed to me the meaning of friendship.

Finally special thanks to my father Prof. Dr Yavuz ERCAN, my mother Nesime ERCAN, my sister İlay İLERİ, my sister`s husband Cihangir İLERİ and My niece Damla İLERİ for their never ending support.

TABLE OF CONTENTS

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Sections

In first section, brief explanation of Artificial Intelligence is given. Today and tomorrow of A.I, areas of Artificial Intelligence and relation between A.I and computer science is mentioned.

In second section, Machine-learning topics are widely pointed out. In addition, the idea of learning and basic learning model are explained.

In third section, which algorithms are used in this thesis and “what are their specifications?” are explained.

In fourth section, software, which is used in this thesis, is clarified.

In fifth section, performance evaluation is explained.

In the last section, conclusion of thesis has been made.

1.2 What Is Artificial Intelligence About?

Artificial Intelligence is a relatively new discipline (born in the middle of the 20th century). It is increasingly frequently mentioned in newspapers, magazines, TV, films and in various kinds of computer entertainments, yet it is not widely understood. Some people think it has already failed and been abandoned, whereas in fact it is steadily growing in academy and industry, though the work is not always labeled as "Artificial Intelligence". That is because some of the important ideas and techniques have been absorbed into software engineering.

It is a central part of one of the most profound scientific and intellectual developments of the last century: the study of information, how it can be acquired, stored, manipulated, extended, used, and transmitted, whether in machines, or humans or other animals. [1]

Physics and chemistry study matter, energy, forces, and the various ways they can be combined and transformed. Biology, geology, medicine, and many other sciences and engineering disciplines build on this by studying more and more complex systems built from physical components. All this research requires an understanding of naturally occurring and artificial machines which operate on forces, energy of various kinds, and transform or rearrange matter.

But some of the machines, both natural and artificial, also manipulate knowledge.

It is now clear that a new kind of science is required for the study of the principles by which

- knowledge is acquired and used,
- goals are generated and achieved,
- information is communicated,
- collaboration is achieved,
- concepts are formed,
- Languages are developed.

We could call it the science of knowledge or the science of intelligence.

This is what AI is about. Not only artificial systems, but also human beings and many living organisms acquire, manipulate, store, use and transmit information. They are also driven or controlled by it: e.g. made happy by praise, made sad by bad news, made afraid by noises in the dark, and so on. From this point it is not surprising that in recent years the study of emotions has been growing in importance in AI.

So AI, despite its unfortunate name, is about natural information processing systems as well as artificial systems, and not just about how they perceive learn and think, but also about what they want and how they feel. It has already had a profound impact on the study of human minds.

How does AI relate to computer science, is it another new discipline? In part it is like the relationship between physics and mathematics. Mathematics develops many concepts and techniques which physics uses, but the central goal of physics is to understand the world, not to understand those techniques. Likewise computer science (along with mathematics, electronic engineering and software engineering) develops general theories about information processing, and helps to create powerful general tools (computers, operating systems and compilers) which are used in AI, but these are not the central focus of AI. The general concepts, techniques and tools produced by computer science are used by AI researchers in the process of studying something else, the kinds of information processing capabilities which we find in many living organisms, and which might also be created in new machines of many kinds.

However, just as the history of physics includes many episodes where mathematics was enriched by the work of theoretical physicists, also AI had a great deal of influence on the development of computer science. But equally it is having a deep impact on other disciplines with which it is connected, especially philosophy, psychology and linguistics.[2],[3]

CHAPTER 2

THE IDEA OF MACHINE LEARNING

The idea behind the learning, how agents work, interaction between the agent and the world is the some important problems in today Artificial Intelligence studies. Machine learning can understand and improve efficiency of human learning. For example, use to improve methods for teaching and tutoring people, as done in Computer-aided instruction. It discovers new things or structure that is unknown to humans. Especially, like in the Data mining. It fills in skeletal or incomplete specifications about a domain.Large, complex AI systems cannot be completely derived by hand and require dynamic updating to incorporate new information. Learning new characteristics expands the domain or expertise. Briefly today's one of the most important topic in the AI is “learning”. Investigating the learning algorithms is a large study area in computer science. [4]

Knowledge Discovery, machine learning is most commonly used to mean the application of induction algorithms, which is one step in the knowledge discovery process. In that definition, training examples are externally supplied, whereas here they are assumed to be supplied by a previous stage of the knowledge discovery process. Machine Learning is the field of scientific study that concentrates on induction algorithms and on other algorithms that can be said to “learn”. It has simply:

Understand and improve efficiency of human learning. For example, use to improve methods for teaching and tutoring people, as done in CAI --- computer-aided instruction

- Discover new things or structure that is unknown to humans Example: Data mining
- Fill in skeletal or incomplete specifications about a domain Large, complex AI systems cannot be completely derived by hand and require dynamic updating to incorporate new information. Learning new characteristics expands the domain or expertise.

2.1 The Basic Learning Model

Learning agents consist of four main components. They are:

Learning element -- the part of the agent responsible for improving its performance

Performance element -- the part that chooses the actions to take

Critic -- tells the learning element how the agent is doing

Problem generator -- suggests actions that could lead to new, informative experiences (suboptimal from the point of view of the performance element, but designed to improve that element)

In designing a learning system, there are four major issues to consider:

components -- which parts of the performance element are to be improved

representation **--** of those components

feedback -- available to the system

prior information -- available to the system

All learning can be thought of as learning the representation of a function.

2.1.1 Types of Learning

There are six main types of learning.

Speed-up learning

A type of deductive learning that requires no additional input, but improves the agent's performance over time. There are two kinds, rote learning and generalization .Data caching is an example of how it would be used.

Learning by taking advice

Deductive learning in which the system can reason about new information added to its knowledge base.

Learning from examples

Inductive learning, concepts are learned from sets of labelled instances.

Clustering

"Natural classes" are found for data instances, as well as ways of classifying them in an unsupervised inductive learning. Examples include COBWEB, AUTOCLASS.

Learning by analogy

A system transfers knowledge from one database into that of a different domain in inductive learning.

Discovery

In this method both inductive and deductive learning in which an agent learns without help from a teacher. It is deductive if it proves theorems and discovers concepts about those theorems; it is inductive when it raises conjectures.

2.2 Topics in Machine Learning

It can be categorized into different approaches:

2.2.1 Artificial Neural Networks

Artificial Neural Network is a system loosely modeled on the human brain. The field goes by many names, such as connectionism, parallel distributed processing, neuro-computing, natural intelligent systems, machine learning algorithms, and artificial neural networks. It is an attempt to simulate within specialized hardware or sophisticated software, the multiple layers of simple processing elements called neurons. Each neuron is linked to certain of its neighbors with varying coefficients of connectivity that represent the strengths of these connections. Learning is accomplished by adjusting these strengths to cause the overall network to output appropriate results.

The most basic components of neural networks are modeled after the structure of the brain. Some neural network structures are not closely to the brain and some does not have a biological counterpart in the brain. However, neural networks have a strong similarity to the biological brain and therefore a great deal of the terminology is borrowed from neuroscience. The most basic element of the human brain is a specific type of cell, which provides us with the abilities to remember, think, and apply previous experiences to our every action. These cells are known as neurons, each of these neurons can connect with up to 200000 other neurons. The power of the brain comes from the numbers of these basic components and the multiple connections between them. Basically, a biological neuron receives inputs from other sources, combines them in some way, performs a generally nonlinear operation on the result, and then output the final result.

The basic unit of neural networks, the artificial neurons, simulates the four basic functions of natural neurons. Of course artificial neurons are much simpler than the biological neuron. Various inputs in the neural network are represented by the mathematical symbol, x (n). Each of these inputs are multiplied by a connection weight, these weights are represented by w (n). In the simplest case, these products are simply summed, fed through a transfer function to generate a result, and then output. Even though all artificial neural networks are constructed from this basic building block the fundamentals may vary in these building blocks and there are differences.

In the design part, the developer must go through a period of trial and error in the design decisions before coming up with a satisfactory design. The design issues in neural networks are complex and are the major concerns of system developers.

<u>Designing a neural network consists of:</u>

* Arranging neurons in various layers.

* Deciding the type of connections among neurons for different layers, as well as among the neurons within a layer.

* Deciding the way a neuron receives input and produces output.

* Determining the strength of connection within the network by allowing the network learns the appropriate values of connection weights by using a training data set.

The process of designing a neural network is an iterative process.

Biologically, layers in neural networks are constructed in a three dimensional way from microscopic components. These neurons seem capable of nearly unrestricted interconnections. This is not true in any man-made network. Artificial neural networks are the simple clustering of the primitive artificial neurons. This clustering occurs by creating layers, which are then connected to one another. How these layers connect may also vary. Basically, all artificial neural networks have a similar structure of topology. Some of the neurons interface the real world to receive its inputs and other neurons provide the real world with the network's outputs. All the rest of the neurons are hidden form view.

Communicating part of NN process like this: Neurons are connected via a network of paths carrying the output of one neuron as input to another neuron. These paths is normally unidirectional, there might however be a two-way connection between two neurons, because there may be another path in reverse direction. A neuron receives input from many neurons, but produces a single output, which is communicated to other neurons. The neuron in a layer may communicate with each other, or they may not have any connections. The neurons of one layer are always connected to the neurons of at least another layer.

INTER-LAYER CONNECTIONS

There are different types of connections used between layers; these connections between layers are called inter-layer connections;

Fully connected

Each neuron on the first layer is connected to every neuron on the second layer.

Partially connected

A neuron of the first layer does not have to be connected to all neurons on the second layer.

Feed forward

The neurons on the first layer send their output to the neurons on the second layer, but they do not receive any input back form the neurons on the second layer.

Bidirectional

There is another set of connections carrying the output of the neurons of the second layer into the neurons of the first layer. Feed forward and bi-directional connections could be fully- or partially connected.

Hierarchical

If a neural network has a hierarchical structure, the neurons of a lower layer may only communicate with neurons on the next level of layer.

Resonance

The layers have bi-directional connections, and they can continue sending Messages across the connections a number of times until a certain condition is achieved. In more complex structures the neurons communicate among themselves within a layer, this is known as intra-layer connections. There are two types of intra-layer connections.

INTRA LAYER CONNECTIONS

Recurrent

The neurons within a layer are fully- or partially connected to one another. After these neurons receive input form another layer, they communicate their outputs with one another a number of times before they are allowed to send their outputs to another layer. Generally, some conditions among the neurons of the layer should be achieved before they communicate their output to another layer.

On-center/Off surround

A neuron within a layer has excitatory connections to itself and its immediate neighbors, and has inhibitory connections to other neurons. One can imagine this type of connection as a competitive gang of neurons. Each gang excites it and its gang members and inhibits all members of other gangs. After a few rounds of signal interchange, the neurons with an active output value will win, and is allowed to update its and its gang member's weights. (There are two types of connections between two neurons, excitatory or inhibitory. In the excitatory connection, the output of one neuron increases the action potential of the neuron to which it is connected. When the connection type between two neurons is inhibitory, then the output of the neuron sending a message would reduce the activity or action potential of the receiving neuron. One causes the summing mechanism of the next neuron to add while the other causes it to subtract. One excites while the other inhibits.

The learning ability of a neural network is determined by its architecture and by the algorithmic method chosen for training.

The training method usually consists of one of three schemes:

Unsupervised

The hidden neurons must find a way to organize themselves without help from the outside. In this approach, no sample outputs are provided to the network against which it can measure its predictive performance for a given vector of inputs. This is learning by doing.

Reinforcement Learning

This method works on reinforcement from the outside. The connections among the neurons in the hidden layer are randomly arranged, then reshuffled as the network is told how close it is to solving the problem. Reinforcement learning is also called supervised learning, because it requires a teacher. The teacher may be a training set of data or an observer who grades the performance of the network results. Both unsupervised and reinforcement suffers from relative slowness and inefficiency relying on a random shuffling to find the proper connection weights.

Backpropogation Network

This method is proven highly successful in training of multilayered neural nets. The network is not just given reinforcement for how it is doing on a task. Information about errors is also filtered back through the system and is used to adjust the connections between the layers, thus improving performance. It is a form of supervised learning.

One can categorize the learning methods into yet another group, *off-line or on-line*. When the system uses input data to change its weights to learn the domain knowledge, the system could be in training mode or learning mode. When the system is being used as a decision aid to make recommendations, it is in the operation mode, this is also sometimes called recall.

Off-line

In the off-line learning methods, once the systems enters into the operation mode, its weights are fixed and do not change any more. Most of the networks are of the off-line learning type.

On-line

In on-line or real time learning, when the system is in operating mode (recall), it continues to learn while being used as a decision tool. This type of learning has a more complex design structure. Neural networks are performing successfully where other methods do not, recognizing and matching complicated, vague, or incomplete patterns. Neural networks have been applied in solving a wide variety of problems.

Most applications of neural networks fall into the following five categories:

Prediction

Uses input values to predict some output. E.g., pick the best stocks in the market, predict weather, and identify people with cancer risk.

Classification

Use input values to determine the classification. E.g., is the input the letter A is the blob of the video data a plane and what kind of plane is it.

Data association

Like classification, it also recognizes data that contains errors. E.g., not only identify the characters that were scanned but identify when the scanner is not working properly.

Data Conceptualization

Analyze the inputs so that grouping relationships can be inferred. E.g., extract from a database the names of those most likely to by a particular product.

Data Filtering

Smooth an input signal. E.g., take the noise out of a telephone signal.

2.2.2 Genetic Algorithms

The idea of applying the biological principle of natural evolution to artificial systems, introduced more than three decades ago, has seen impressive growth in the past few years. Usually grouped under the term evolutionary algorithms or evolutionary computation, it was found the domains of genetic algorithms, evolution strategies, evolutionary programming, and genetic programming. Evolutionary algorithms are everywhere nowadays, having been successfully applied to numerous problems from different domains, including optimization, automatic programming, machine learning, economics, operations research, ecology, and population genetics, studies of evolution and learning, and social systems.

A genetic algorithm is an iterative procedure that consists of a constant-size population of individuals, each one represented by a finite string of symbols, known as the genome, encoding a possible solution in a given problem space. This space, referred to as the search space, comprises all possible solutions to the problem at hand. Generally speaking, the genetic algorithm is applied to spaces which are too large to be fully searched. The symbol alphabet used is often binary, though other representations have also been used, including character-based encodings, real-valued encodings, and -- most remarkably -- tree representations.

The standard genetic algorithm proceeds as follows: an initial population of individuals is generated at random or heuristically. Every evolutionary step, known as a generation, the individuals in the current population are decoded and evaluated according to some predefined quality criterion, referred to as the fitness, or fitness function. To form a new population, individuals are selected according to their fitness. Many selection procedures are currently in use, one of the simplest being Holland's original fitness-proportionate selection, where individuals are selected with a probability proportional to their relative fitness. This ensures that the expected number of times an individual is chosen is approximately proportional to its relative performance in the population. Thus, high-fitness (``good'') individuals stand a better chance of ``reproducing'', while low-fitness ones are more likely to disappear.

Selection alone cannot introduce any new individuals into the population, specifically; it cannot find new points in the search space. These are generated by genetically-inspired operators, of which the most well known are crossover and mutation. Crossover is performed with probability pcross (the ``crossover probability'' or ``crossover rate'') between two selected individuals, called parents, by exchanging parts of their genomes (i.e., encodings) to form two new individuals, called offspring; in its simplest form, substrings are exchanged after a randomly selected crossover point. This operator tends to enable the evolutionary process to move toward ``promising'' regions of the search space. The mutation operator is introduced to prevent premature convergence to local optima by randomly sampling new points in the search space. It is carried out by flipping bits at random, with some (small) probability pmut. Genetic algorithms are stochastic iterative processes that are not guaranteed to converge; the termination condition may be specified as some fixed maximal number of generations or as the attainment of an acceptable fitness level. Below presents the standard genetic algorithm in pseudo-code format:

```
Begin GA
 g:=0  { generation counter }
 Initialize population P(g)
 Evaluate population P(g)  {i.e., compute fitness values }
 While not done do
  g:=g+1
  Select P(g) from P(g-1)
  Crossover P(g)
```

```
    Mutate P(g)
    Evaluate P(g)
  end while
end GA
```

The implementation of an evolutionary algorithm, an issue which usually remains in the background, is quite costly in many cases, since populations of solutions are involved possibly coupled with computation-intensive fitness evaluations. One possible solution is to parallelize the process, an idea that has been explored to some extent in recent years. While posing no major problems in principle, this may require cautious modifications of existing algorithms or the introduction of new ones in order to meet the constraints of a given parallel machine.

2.2.3 Inductive Learning

Inductive learning is a kind of learning in which, given a set of examples an agent tries to estimate or create an evaluation function. Most inductive learning is supervised learning, in which examples provided with classifications. (The alternative is clustering.) More formally, an example is a pair (x, f(x)), where x is the input and f(x) is the output of the function applied to x. The task of pure inductive inference (or induction) is, given a set of examples of f, to find a hypothesis h that approximates f.

2.2.3.1 Decision Trees

A decision tree is a simple inductive learning structure. Given an instance of an object or situation, which is specified by a set of properties, the tree returns a "yes" or "no" decision about that instance. Each internal node in the tree represents a test on one of those properties, and the branches from the node are labelled with the possible outcomes of the test. Each leaf node is a Boolean classifier for the input instance.

From a logical viewpoint, decision trees are attempting to learn sets of implication sentences; this set is called the goal predicate. Note that these implications can only test propositions on the input object. However, any Boolean function can be represented as a decision tree.

Decision trees very inefficiently represent some Boolean functions. The parity function is one example; it requires an exponentially large tree, since it must test all of the attributes. Another difficult function is the majority function. (It could be characterizing this kind of function roughly as one that cannot be determined on the basis only of parts of the input; the problem is that the whole thing has to be examined. The whole point of the decision tree is to find ways that only parts of the input need to be examined in order to reach a decision).The point of the decision tree algorithm is to find a "smallest" tree that correctly classifies most of the training examples.

Algorithm

The decision tree algorithm works roughly as follows:

If there are no examples to split on, return a default classification.

If all examples have the same classification, return it.

Otherwise, choose the property with the highest information gain:

Create a new decision tree rooted on a test of that property.

```
For each possible value of that property:
  Recursively call this algorithm with the subset of examples that
    Match the property on the new tree, and their classification
    (By majority vote in case of conflict).
  Add a branch to the sub tree pointing to the result.
Return this tree.
```

The approach is to test the attribute with the highest information gain first, and then those with successively smaller gains, until all examples are classified.

Two problems arise in this induction. If a split leads to a branch with no existing examples, its classification is given by some preset default. If a set of identical examples has different classifications (noise), then some method must be used to determine what classification to use; majority-vote is a simple one.

Extensions of Basic Decision Tree

There are several ways to expand upon the trees described. One is to allow them to handle missing data, where the values of some attributes in the test set are not known. Several approaches have been suggested: assign a value based upon the distribution of values in other instances, or construct another decision tree just to decide the value. The simplest is to give it the most common value, and in practice this seems to produce nearly as good results as the other methods.

A second extension is to allow multi-valued attributes, beyond Boolean. Yet another is to permit continuous-valued attributes, which would need to be discredited in some way to be used in a decision tree. Another is allowing more than two possible classifications; ID3 algorithm does not do this, but later systems do.

Two problems with ID3 that were later solved by Quinlan: the smaller the sub tree gets the harder to intelligently choose an attribute to split on and decision trees cannot represent DNF concepts efficiently. These problems were solved by converting the tree into production rules of the form

f_1 ^ f_2 ^ ... f_k -> class.

Statistically insignificant conditions from the left-hand side are removed, and then entire rules that do not significantly improve performance (determined by comparing classification using rule sets with and without the rule under consideration) are deleted. This can both improve accuracy and dramatically reduce the complexity of the learned concept.

ID5 is an improved version of ID3 that supports incremental learning. It processes the examples one at a time and produces an updated decision tree after each example. The basic idea of growing the tree top-down remains the same, but as the examples are moved through the tree, they are stored at the leaves. If a leaf node comes to have both positive and negative examples, a new sub tree is created at that node, splitting on the attribute with the highest information gain for those examples.

2.2.4 Concept Learning and the General-To-Specific Ordering

Concept Learning is the process of inferring a Boolean-valued function from training examples of its input and output.

* Learning from examples

* General-to-specific ordering over hypotheses

* Version spaces and candidate elimination algorithm

* Picking new examples

* The need for inductive bias

- It is concerned with acquiring the definition of a general category (concept) from a sample of positive and negative training examples of the category.

- It can be formulated as a problem of searching through a predefined space of potential hypotheses for one that best fits the training examples.

- The search can be efficiently organized by general-to-specific ordering of hypotheses.

Algorithms: Find-S and Version Space.

- The main issue: inductive bias

The topics in concept learning are:

- A concept learning task

- Concept learning as search

- Find-S: Finding a maximally specific hypothesis

- Version spaces

* The candidate elimination algorithm

* The boundary set representation

- Inductive bias

- Concept learning can be cast as a search problem. General-to-specific ordering of hypotheses provides a useful search structure. The version space approach is good for single concept learning.

2.2.5 Evaluating Hypotheses

The importance of evaluating hypothesis is to understand whether to use the hypothesis and it is an integral component of learning algorithm. But it has two difficulties with only set of limited data.

1. Bias in Estimate

Use test examples chosen independently of the training examples.

2. Variance in Estimate

Use larger set of test examples.

Summary of evaluating hypothesis like below:

• Statistical theory provides a basis for estimating true error of a hypothesis h

• One possible cause of errors in estimating accuracy is estimation bias

• A second cause is variance in estimate

• Confidence intervals provide useful bounds for error

• Holdout methods provide an affective method for training and testing on limited data

• Statistical models provide approximate confidence intervals that can be of great help in interpreting experimental comparisons of learning methods

2.2.6 Bayesian Learning

Learning that treats the problem of building hypotheses as a particular case of the problem of making predictions. The probabilities of various hypotheses are estimated, and predictions are made using the posterior probabilities of the hypotheses to weight them. This is a probabilistic approach to learning and inference. It is based on the assumption that the quantities of interest are governed by probability distributions. It is attractive because in theory it can arrive at optimal decisions. It provides a quantitative approach to weighing the evidence supporting alternative hypotheses. Conventional training methods for multilayer perceptrons ("backpropagated" nets) can be interpreted in statistical terms as variations on maximum chances estimation. The idea is to find a single set of weights for the network that maximize the fit to the training data, perhaps modified by some sort of weight penalty to prevent over fitting. The Bayesian school of statistics is based on a different view of what it means to learn from data, in which probability is used to represent uncertainty about the relationship being learned. Before we have seen any data, our prior opinions about what the true relationship might be can be expressed in a probability distribution over the network weights that define this relationship. After looking at the data our revised opinions are captured by a posterior distribution over network weights. Network weights that seemed plausible before, but which don't match the data very well, will now be seen as being much less likely, while the probability for values of the weights that do fit the data well will have increased.

Typically, the purpose of training is to make predictions for future cases in which only the inputs to the network are known. The result of conventional network training is a single set of weights that can be used to make such predictions. In contrast, the result of Bayesian training is a posterior distribution over network weights. If the inputs of the network are set to the values for some new case, the posterior distribution over network weights will give rise to a distribution over the outputs of the network, which is known as the predictive distribution for this new case. If a single-valued prediction is needed, one might use the mean of the predictive distribution, but the full predictive distribution also tells you how uncertain this prediction is. Why bother with all this? The hope is that Bayesian methods will provide solutions to such fundamental problems as:

* How to judge the uncertainty of predictions. This can be solved by looking at the predictive distribution, as described above.

* How to choose appropriate network architecture (e.g., the number hidden layers, the number of hidden units in each layer).

* How to adapt to the characteristics of the data (e.g., the smoothness of the function, the degree to which different inputs are relevant).

Good solutions to these problems, especially the last two, depend on using the right prior distribution, one that properly represents the uncertainty that you probably have about which inputs are relevant, how smooth the function is, how much noise there is in the observations, etc. Such carefully vague prior distributions are usually defined in a hierarchical fashion, using hyper parameters, some of which are analogous to the weight decay constants of more conventional training procedures. The use of hyperparameters is discussed in particular use of an "Automatic Relevance

Determination" scheme that aims to allow many possibly-relevant inputs to be included without damaging effects. Selection of appropriate network architecture is another place where prior knowledge plays a role. One approach is to use a very general architecture, with lots of hidden units, maybe in several layers or groups, controlled using hyper parameters. It is also possible to choose between architectures in a Bayesian fashion, using the "evidence" for architecture. Implementing all this is one of the biggest problems with Bayesian methods. Dealing with a distribution over weights (and perhaps hyper parameters) is not as simple as finding a single "best" value for the weights. Exact analytical methods for models of as complex as neural networks are out of the question. Two approaches have been tried:

1. Find the weights/hyper parameters that are most probable, using methods similar to conventional training (with regularization), and then approximate the distribution over weights using information available at this maximum.

2. Use a Monte Carlo method to sample from the distribution over weights. The most efficient implementations of this use dynamical Monte Carlo method whose operation resembles that of backprop with momentum.

Work on Bayesian neural network learning has so far concentrated on multilayer perceptron networks, but Bayesian methods can in principal be applied to other network models, as long as they can be interpreted in statistical terms. For some models (RBF networks), this should be a fairly simple matter; for others (Boltzmann Machines), substantial computational problems would need to be solved.

2.2.7 Computational Learning Theory

Computational Learning Theory analyses the sample complexity and computational complexity of inductive learning. There is a trade-off between the expressiveness of the hypothesis language and the ease of learning. It is the theory about learn ability, learning error, sample size and computational complexity.[6]

What general laws constrain inductive learning?

- Probability of successful learning
- Number of training examples
- Complexity of hypothesis space
- Accuracy to which target concept is approximated
- Manner in which training examples presented

How can one possibly know that one's learning algorithm has produced a theory that will correctly predict the future? In terms of the definition of inductive learning, how do we know that the hypothesis h is close to the target function f if we don't know what f is?

Let's focus on the answers provided by Computational Learning Theory, a field at the intersection of AI and theoretical computer science.

The underlying principle is the following: any hypothesis that is seriously wrong will almost certainly be "found out" with high probability after a small number of examples, because it will make an incorrect prediction. Thus, any hypothesis that is inconsistent with a sufficiently large set of training examples is unlikely to be seriously wrong that is, it must be Probably Approximately Correct. PAC learning is the subfield of computational learning theory that is devoted to the idea.

2.2.8 Cased-Based Learning

Case-Based Reasoning is one of most successful applied AI technologies of recent years. Commercial and industrial applications can be developed rapidly and existing corporate databases can be used as knowledge sources. Helpdesks and diagnostic systems are the most common applications.

Case-Based Reasoning (CBR) is based on the intuition that new problems are often similar to previously encountered problems, and therefore, that past solutions may be of use in the current situation. Case-Based Reasoners typically apply retrieval algorithms and matching algorithms to a case base of past problem-solution pairs. Cases are often derived from legacy databases and need not be well structured: case-based reasoning is robust and requires little knowledge acquisition. In complex applications such as planning and design it is insufficient to recall the best matching cases---cases must be adapted to form a new solution.

CBR has been applied to classification tasks, e.g. to determine the type of an organism from observed attributes and to determine whether or not cancer treatment is necessary given past cases. CBR has also been applied to design tasks, e.g. the optimal layout of items in furnace. The major application of CBR is helpdesks. The helpdesk operator can enter the client's description of their problem into the CBR system, for example, the client may describe a malfunction of their PC, and will receive the most appropriate past solutions from the corporate database as a result.

CBR techniques can be applied to large and complex datasets. Data points are considered to be cases, and may be described by tens, or even hundreds, of attributes. One attribute will be used as the diagnostic goal, e.g. faulty or operational, and the CBR will attempt to partition a training set of data with maximum accuracy. Critical factors in the success of this type of CBR are the amount of data available for the system to learn from, and the existence of particular properties in the data itself.

2.2.9 Learning Sets of Rules

Learning rule sets is like learning disjunctive concepts. Learning rules involving variables is challenging. If-then rules are one of the most expressive representations of knowledge.

- Learn sets of rules by using ID3 and then converting the tree to rules.
- Use a genetic algorithm that encodes the rules as bit strings.
- But, these only work with predicate rules (no variables).
- They also consider the set of rules as a whole, not one rule at a time.

Rules

- First-order predicate logic (calculus) formalizes statements using predicates (Boolean functions) and functions. Both can have variables.
- A rule set can look like

IF Parent(x,y) THEN Ancestor(x,y)

IF Parent(x,z) Ancestor(z,y) THEN Ancestor(x,y)

- Here, Parent(x,y) is a predicate that indicates that y is the parent of x.

- These two rules form a recursive function which would be very hard to represent using a decision tree or propositional representation.

- In Prolog, programs are set of first-order rules with the form as above (known as Horn clauses).

- View the learning of rules as the learning of Prolog programs.

Sequential Covering

- The idea in a sequential covering algorithm is to learn one rule, remove the data it covers, and then repeat.

Comparison on Rule Learning Methods:

Symbolic heuristic search:

This method commonly uses general to specific beam search or hill-climbing search. A performance criterion based on accuracy and coverage needs to be defined for evaluating and selecting rules. However, this criterion is often ill-defined especially in the case of noise, inconsistency, and uncertainty. And there is no good theoretical guidance for global optimization.

Decision trees:

In this approach, classification knowledge is first represented as a decision tree and then the tree is translated as a set of rules. The decision tree is constructed by sequentially selecting attributes based on an information theoretical measure. This approach has the advantage in speed but it searches incompletely through a complete hypothesis space and is also sensitive to data noise.

Inverted logic deduction:

In this approach, learning proceeds by generating a hypothesis that, together with some background knowledge, explains the given data. This approach does not naturally handle noise, inconsistency, and uncertainty. The search through the hypothesis space is intractable in the general case and increasingly complex with the amount of background knowledge. So far, there is no good solution to all of these problems together.

Neural networks:

In this approach, a neural network learns a function to fit the given data, and then the function is decoded as a set of rules. There is good theoretical support in functional approximation, but what remains to be solved is how to extract correct rules from a trained neural network.

Genetic Algorithms:

In this approach, each rule set is encoded as a bit string and genetic search operators are applied to explore the hypothesis space. The stochastic nature of the algorithm provides a means for alleviating the local minima effect, but the element of randomness may also introduce some degree of imprecision. Experience has shown that this approach fails to learn true domain rules even in not too complex domains.

2.2.10 Analytical Learning

* Using background knowledge to explain (prove) training example is member of the target concept.

* Then, generalize explanation

* ILP used background knowledge to suggest new features (i.e., increase H)

* Analytical learning uses BK to decrease (constrain) H

* Prolog -EBG

* Perfect vs. imperfect domain theories

2.2.11 Combining Inductive and Analytical Learning

Table 1 – Inductive and Analytical Learning

	Inductive Learning	**Analytical Learning**
Goal	Hypothesis fits data	Hypothesis fits domain theory
Justification	Statistical inference	Deductive Inference
Advantages	Requires little prior knowledge	Learns from scarce data
Pitfalls	Scarce data, incorrect bias	Imperfect domain theory

- They seem very complementary.

It is wanted a learning method such that:

1. Given no domain theory it should be as good as purely inductive methods.

2. Given a perfect domain theory it should be as good as analytical methods.

3. Given imperfect domain theory and imperfect data it should combine the two and do batter than both inductive and analytical.

4. Accommodate an unknown level of error in training data.

5. Accommodate an unknown level of error in domain theory.

It is purely inductive;

* need no explicit prior knowledge

* fail when scarce data

* may be misled by wrong bias

It is purely analytical;

* learn even from scarce data

* fail when incorrect or insufficient knowledge

Both of them have justifications:

Inductive;

* Statistical justification

* Hypothesis fits statistically data

Analytical;

* Logical justification

* Hypothesis fits domain theory & data

2.2.12 Reinforcement Learning

Reinforcement learning is learning what to do---how to map situations to actions---so as to maximize a numerical reward signal. The learner is not told which actions to take, as in most forms of machine learning, but instead must discover which actions yield the most reward by trying them. In the most interesting and challenging cases, actions may affect not only the immediate reward, but also the next situation and, through that, all subsequent rewards. These two characteristics---trial-and-error search and delayed reward---are the two most important distinguishing features of reinforcement learning.

Reinforcement learning is defined not by characterizing learning algorithms, but by characterizing a learning problem. Any algorithm that is well suited to solving that problem we consider to be a reinforcement learning algorithm. The basic idea is simply to capture the most important aspects of the real problem facing a learning agent interacting with its environment to achieve a goal. Clearly such an agent must be able to sense the state of the environment to some extent and must be able to take actions that affect that state. The agent must also have a goal or goals relating to the state of the environment. Our formulation is intended to include just these three aspects---sensation, action, and goal---in the simplest possible form without trivializing any of them.

Reinforcement learning is different from supervised learning, the kind of learning studied in most current research in machine learning, statistical pattern recognition, and artificial neural networks. Supervised learning is learning from examples provided by some knowledgeable external supervisor. This is an important kind of learning, but alone it is not adequate for learning from interaction. In interactive problems it is often impractical to obtain examples of desired behavior that are both correct and representative of all the situations in which the agent has to act. In uncharted territory---where one would expect learning to be most beneficial---an agent must be able to learn from its own experience.

One of the challenges that arise in reinforcement learning and not in other kinds of learning is the tradeoff between exploration and exploitation. To obtain a lot of reward, a reinforcement learning agent must prefer actions that it has tried in the past and found to be effective in producing reward. But to discover such actions it has to try actions that it has not selected before. The agent has to exploit what it already knows in order to obtain reward, but it also has to explore in order to make better action selections in the future. The dilemma is that neither exploitation nor exploration can be pursued exclusively without failing at the task. The agent must try a variety of actions and progressively favor those that appear to be best. On a stochastic task, each action must be tried many times to reliably estimate its expected reward.

Another key feature of reinforcement learning is that it explicitly considers the whole problem of a goal-directed agent interacting with an uncertain environment. This is in contrast with many approaches that address sub problems without addressing how they might fit into a larger picture.

Reinforcement learning takes the opposite tack, by starting with a complete, interactive, goal-seeking agent. All reinforcement learning agents have explicit goals, can sense aspects of their environments, and can choose actions to influence their environments. Moreover, it is usually assumed from the beginning that the agent has to operate despite significant uncertainty about the environment it faces. When reinforcement learning involves planning, it has to address the interplay between planning and real-time action selection, as well as the question of how environmental models are acquired and improved. When reinforcement learning involves supervised learning, it does so for very specific reasons that determine which capabilities are critical, and which are not. For learning research to make progress, important subproblems have to be isolated and studied, but they should be subproblems that are motivated by clear roles in complete, interactive, goal-seeking agents, even if all the details of the complete agent cannot yet be filled in.

Elements of Reinforcement Learning

Beyond the agent and the environment, one can identify four main sub-elements to a reinforcement learning system: a policy, a reward function, a value function, and, optionally, a model of the environment.

A policy defines the learning agent's way of behaving at a given time. Roughly speaking, a policy is a mapping from perceived states of the environment to actions to be taken when in those states. It corresponds to what in psychology would be called a set of stimulus-response rules or associations. In some cases the policy may be a simple function or lookup table, whereas in others it may involve extensive computation such as a search process. The policy is the core of a reinforcement learning agent in the sense that it alone is sufficient to determine behavior. In general, policies may be stochastic.

A reward function defines the goal in a reinforcement learning problem. Roughly speaking, it maps perceived states (or state-action pairs) of the environment to a single number, a reward, indicating the intrinsic desirability of the state. A reinforcement-learning agent's sole objective is to maximize the total reward it receives in the long run. The reward function defines what the good and bad events are for the agent. In a biological system, it would not be inappropriate to identify rewards with pleasure and pain. They are the immediate and defining features of the problem faced by the agent. As such, the reward function must necessarily be fixed. It may, however, be used as a basis for changing the policy. For example, if an action selected by the policy is followed by low reward then the policy may be changed to select some other action in that situation in the future. In general, reward functions may also be stochastic.

Whereas a reward function indicates what is good in an immediate sense, a value function specifies what is good in the long run. Roughly speaking, the value of a state is the total amount of reward an agent can expect to accumulate over the future starting from that state. Whereas rewards determine the immediate, intrinsic desirability of environmental states, values indicate the long-term desirability of states after taking into account the states that are likely to follow, and the rewards available in those states. For example, a state might always yield a low immediate reward, but still have a high value because it is regularly followed by other states that yield high rewards. Or the reverse could be true. To make a human analogy, rewards are like pleasure (if high) and pain (if low), whereas values correspond to a more refined and far-sighted judgment of how pleased or displeased we are that our environment is in a particular state. Expressed this way, we hope it is clear that value functions formalize a very basic and familiar idea.

Rewards are in a sense primary, whereas values, as predictions of rewards, are secondary. Without rewards there could be no values, and the only purpose of estimating values is to achieve more reward. Nevertheless, it is values with which we are most concerned when making and evaluating decisions. Action choices are made on the basis of value judgments. Seeking actions that bring about states of highest value is what is wanted, not highest reward, because these actions obtain for us the greatest amount of reward over the long run. In decision-making and planning, the derived quantity called value is the one with which we are most concerned. Unfortunately, it is also much harder to determine values than it is to determine rewards. Rewards are basically given directly by the environment, but values must be estimated and re-estimated from the sequences of observations an agent makes over its entire lifetime. In fact, the most important component of almost all reinforcement learning algorithms is a method for efficiently estimating values. The importance and centrality of estimating values is perhaps the most important thing we have learned about reinforcement learning in the last two decades.

The fourth and final element of some reinforcement learning systems is a model of the environment. This is something that mimics the behavior of the environment. For example, given a state and action, the model might predict the resultant next state and next reward. Models are used for planning, by which we mean any way of deciding on a course of action by considering possible future situations before they are actually experienced. The incorporation of models and planning into reinforcement learning systems is a relatively new development. Early reinforcement learning systems were explicitly trial-and-error learners; what they did was viewed as almost the opposite of planning. Nevertheless, it gradually became clear that reinforcement learning methods are closely related to dynamic programming methods, which do use models, and that they in turn are closely related to state-space planning methods.

Modern reinforcement learning spans the spectrum from low-level, trial-and-error learning to high-level, deliberative planning.[10]

CHAPTER 3

ALGORITHMS

3.1 C4.5 Algorithm

C4.5 is a later version of the ID3 decision tree induction algorithm. C4.5 introduces a number of extensions of the original ID3 algorithm. In building a decision tree we can deal with training sets that have records with unknown attribute values by evaluating the gain, or the gain ratio, for an attribute by considering only the records where that attribute is defined. In using a decision tree, we can classify records that have unknown attribute values by estimating the probability of the various possible results.

C4.5 is a decision tree that allows both discrete and continuous input variables. The way in which the input data is organized can be chosen column by column and allows the threshold splitting similar to CART as well as the other methods found in CHAID. However, in deciding where the splits are made it uses an 'information gain' criterion. It is possible to choose the selected processing method (threshold, static, k-means or normal) across all columns or by column. Care needs to be taken to ensure that the processing method chosen matches the form of attribute data being used, i.e. threshold on continuous data, buckets and clustering on either and normal only on discrete data

3.2 Cart Algorithm

CART is a decision tree that treats all data inputs as continuous variables and splits the data by using thresholds. Values of an attribute (column) that are above the threshold split one way and those below the other. The split is made on the maximum membership of a target class and non-membership of other target classes. Only binary trees (those splitting two ways from a node) are produced. CART cannot handle categorical data with more than two categories in an attribute.

3.3 Quest Algorithm

QUEST stands for “Quick, Unbiased, and Efficient Statistical Trees” and is a program for tree-structured classification. The algorithms are described in Loh and Shih (1997). The performance of QUEST compared with other classification methods can be found in Lim, Loh and Shih (2000). The main strengths of QUEST are unbiased variable selection and fast computational speed. In addition, it has options to perform CART-style exhaustive search and cost-complexity cross-validation pruning. [12]

3.4 Chaid Algorithm

CHAID is a decision tree that treats all data inputs as discrete variables or members of a class and splits the data according to the number of classes. Splits are made using a chi-square statistical significance test. There are three processing methods available to categorize and condition the input data and the choice can be made across all the attributes (columns) or on a per column basis. The choices are:

Static: This splits the data range of attribute values for the column chosen into sets of equal size. The number of 'buckets' is selected by entering the number chosen in the grouping section of the dialog (having de-selected 'using auto-setting').

K-Means: This splits the data range of attribute values for the column chosen into sets of a size selected by K-means pre-processing of the column data. The number of K-means clusters is selected by entering the number chosen in the grouping section of the dialog (having de-selected 'using auto-setting'). The thresholds in the column data will be mid way between the centers of the clusters found.

Normal: This splits the data such that there is an input class for each attribute value in the column. This should only be used with discrete data inputs, however, in the case of continuous data the threshold values will be mid way between each attribute value.

3.5 Id3 Algorithm

Very simply, ID3 builds a decision tree from a fixed set of examples. The resulting tree is used to classify future samples. The example has several attributes and belongs to a class (like yes or no). The leaf nodes of the decision tree contain the class name whereas a non-leaf node is a decision node. The decision node is an attribute test with each branch (to another decision tree) being a possible value of the attribute. ID3 uses information gain to help it decide which attribute goes into a decision node. The advantage of learning a decision tree is that a program, rather than a knowledge engineer, elicits knowledge from an expert.

ID3 is a nonincremental algorithm, meaning it derives its classes from a fixed set of training instances. An incremental algorithm revises the current concept definition, if necessary, with a new sample. The classes created by ID3 are inductive, that is, given a small set of training instances, the specific classes created by ID3 are expected to work for all future instances. The distribution of the unknowns must be the same as the test cases. Induction classes cannot be proven to work in every case since they may classify an infinite number of instances. Note that ID3 (or any inductive algorithm) may misclassify data.

Data description:

The sample data used by ID3 has certain requirements, which are:

Attribute-value description - the same attributes must describe each example and have a fixed number of values.

Predefined classes - an example's attributes must already be defined, that is, they are not learned by ID3.

Discrete classes - classes must be sharply delineated. Continuous classes broken up into vague categories such as a metal being "hard, quite hard, flexible, soft, quite soft" are suspect.

Sufficient examples - since inductive generalization is used (i.e. not provable) there must be enough test cases to distinguish valid patterns from chance occurrences.

Attribute Selection:

How does ID3 decide which attribute is the best? A statistical property, called information gain, is used. Gain measures how well a given attribute separates training examples into targeted classes. The one with the highest information (information being the most useful for classification) is selected. In order to define gain, we first borrow an idea from information theory called entropy. Entropy measures the amount of information in an attribute.

Given a collection S of c outcomes

Entropy(S) = Sum -p (I) log2 p (I)

Where p (I) is the proportion of S belonging to class I. Sum is over c. Log2 is log base 2.

Note that S is not an attribute but the entire sample set.

3.6 Comparison of Classification Tree Methods

FEATURE	QUEST	CART	CHAID	C4.5	ID3
Split variable selection					
Unbiased selection	Y	N	N	N	N
Split types					
Univariate (axis-orthogonal)	Y	Y	Y	Y	Y
Linear combinations (oblique)	Y	Y	N	N	N
Choice of misclassification costs	Y	Y	Y	N	N

Choice of class prior probabilities	Y	Y	N	N	N
Choice of impurity functions	Y	Y	N	Y	Y
Bagging	N	Y	N	N	N
Error estimation by					
cross-validation	Y	Y	Y	N	N
Number of branches at each node					
Always two	Y	Y	N	N	N
Two or more	N	N	Y	Y	Y
Missing value methods					
Imputation	Y	N	N	N	N
Alternate/surrogate splits	N	Y	N	N	N
Missing value branch	N	N	Y	N	N
Probability weights	N	N	N	Y	Y
Tree size control					
Stopping rule	N	N	Y	N	N
Pre-pruning	N	N	N	Y	Y
Test-sample pruning	Y	Y	N	N	N
Cross-validation pruning	Y	Y	N	N	N

Tree diagram formats	
Text	Y Y Y Y Y
LATEX	Y Y N N N
allClear	Y N N N N
Proprietary	N Y Y N N
Platforms	
Windows	Y Y Y Y Y
Linux	Y Y N Y Y
Sun	Y Y N Y N

CHAPTER 4

PROGRAMS

4.1 SIPINA (Research Edition)

SIPINA is software which can extract knowledge from data.

SIPINA_W© learns from quantitative and qualitative data. It produces a lattice graph. The trees are a particular case of a lattice graph. SIPINA method is more general than induction trees like C4.5, ID3, and CHAID...

In this program, the lattice graph issued from the learning step is translated in terms of production rules and stored in a Knowledge Base System (KBS). SIPINA_W© analyses the rules and detects several anomalies such as redundancy, contradictions, and cancels them. SIPINA_W© can merge many KBS's and optimize the final KBS.

The validation of the learning is performed via an inference engine. For that, first you choose a data file and a KBS and then SIPINA_W© predicts the membership class of the examples in the file. In the same manner, the generalization is done on any other file.

Furthermore, you may execute cross-validations and, when working with some analysis methods, it is possible to use the pruning techniques concerning the induction tree. Moreover, for some methods you are able to Use the ‘stop growing’ technique on the construction of the graph. [13]

CHAPTER 5

ALGORITHM SELECTION

5.1 Selection

Choosing the best algorithm for a dataset was needed a broad survey on machine learning. We studied and investigated on learning and learning algorithms in order to make the right decision. After surveys, which we were done, finished, we agreed to select the decision tree algorithms. These algorithms are QUEST, CHAID, ID3, C4.5 and CART.

5.2 Executing Algorithms

These algorithms have many different features. To decide which parameters are suitable for our dataset is vary from person to person or dataset. We chose the parameters below:

QUEST was executed with conditions; Estimated priors, minimal node size 5,10 or 15, use univariate split, use(unbiased)statistical tests for variable selection, alpha value(0.50), splint point method(discriminant analysis), for categorical split point use CRIMCOORD and QDA, use 10-fold CV Sample pruning CART was also executed with conditions same as QUEST.

C4.5 was executed with specifications; C.L for pessimistic pruning = 25, size of leaves = 2, sampling = All Dataset and size of sample is % 50 with random sampling.

ID3 was executed with conditions; Confidence level = 0.50, sampling = All Dataset and size of sample is % 50 with random sampling.

CHAID was executed with conditions; P-level for merging nodes = 0.50 and for splitting nodes = 0.00001, bonferroni adjustments is manual (1), sampling = All Dataset and size of sample is % 50 with random sampling., SE-rule trees based on number of SEs = 1.00.

5.2.1 Quest

QUEST showed that it was not an efficient algorithm for our dataset. It has the lowest accuracy for classified rules (0.5507). Moreover, it was referring that when the minimal node size increased, the misclassification cost for confusion matrix decreased. For minimal node size = 5, cost was 0.4963. For minimal node size = 10, cost was 0.4814. For minimal node size = 15, cost was 0.4781. The rules for QUEST:

Node Left node Right node   Split variable   Predicted class

```
 1       2        3       maxsıcaklık

 2       4        5       maxsıcaklık

 4  * terminal node *                   (601000-700000)

 5  * terminal node *                   (701000-740000)

 3      88       89       minsıcaklık

88  * terminal node *                   (741000-800000)

89  * terminal node *                   (801000-900000)
```

Classification tree :

```
   Node 1: maxsıcaklık <= 16.74

     Node 2: maxsıcaklık <= 9.519

       Node 4: (601000-700000)

     Node 2: maxsıcaklık > 9.519

       Node 5: (701000-740000)

   Node 1: maxsıcaklık > 16.74

     Node 3: minsıcaklık <= 12.22

       Node 88: (741000-800000)

     Node 3: minsıcaklık > 12.22

       Node 89: (801000-900000)
```

5.2.2 Chaid

CHAID was the least accurate algorithm for our dataset. It has accuracy = 0.5330.The average misclassification cost for confusion matrix was 0.4395 for 5 times execution. It has the rules below:

For one execution:

IF maxsıc >=22.30 and minsıc < 13.55 THEN sutüket in [(741000-800000)] with accuracy 0.5000 on (18,1,1,0,13,2,1)

IF maxsıc >=22.30 and minsıc >=13.55 THEN sutüket in [(801000-900000)] with accuracy 0.6458 on (0,0,2,0,31,15,0)

IF maxsıc < 22.30 and maxsıc < 14.00 THEN sutüket in [(601000-700000)] with accuracy 0.4688 on (5,29,30,0,0,0,0)

IF maxsıc < 22.30 and maxsıc >=14.00 THEN sutüket in [(741000-800000)] with accuracy 0.5294 on (18,9,6,0,1,0,0)

5.2.3 Cart

CART has the accuracy = 0.5863.It was the third accurate algorithm. Its misclassification cost increased when the minimal node size increased. Its rule table:

Node Left node Right node Split variable Predicted class

```
 1      2      3      maxsıcaklık

 2      4      5      maxsıcaklık

 4      6      7      maxsıcaklık

 6  * terminal node *                   (601000-700000)

 7  * terminal node *                   (701000-740000)

 5     58     59      minsıcaklık

58  * terminal node *                   (741000-800000)

59  * terminal node *                   (701000-740000)

 3     98     99      minsıcaklık

98  * terminal node *                   (741000-800000)

99  * terminal node *                   (801000-900000)

Classification tree:

   Node 1: maxsıcaklık <= 16.74

    Node 2: maxsıcaklık <= 9.519

     Node 4: maxsıcaklık <= 6.020

      Node 6: (601000-700000)

     Node 4: maxsıcaklık > 6.020

      Node 7: (701000-740000)
```

Node 2: maxsıcaklık > 9.519

Node 5: minsıcaklık <= -0.3867

Node 58: (741000-800000)

Node 5: minsıcaklık > -0.3867

Node 59: (701000-740000)

Node 1: maxsıcaklık > 16.74

Node 3: minsıcaklık <= 12.22

Node 98: (741000-800000)

Node 3: minsıcaklık > 12.22

Node 99: (801000-900000)

5.2.4 ID3

This decision tree algorithm has the second best misclassification cost for confusion matrix which was 0.3527 for 5 times execution. And it has the second best accuracy which was 6813.It has rules below:

For one execution:

IF maxsıc >=22.30 and minsıc < 13.55 THEN sutüket in [(741000-800000)] with accuracy 0.5000 on (18,1,1,0,13,2,1)

IF maxsıc >=22.30 and minsıc >=13.55 THEN sutüket in [(801000-900000)] with accuracy 0.6458 on (0,0,2,0,31,15,0)

IF maxsıc < 22.30 and maxsıc >=14.00 and yağış < 0.05 THEN sutüket in [(601000-700000)] with accuracy 1.0000 on (0,0,3,0,0,0,0)

IF maxsıc < 22.30 and maxsıc >=14.00 and yağış >=0.05 THEN sutüket in [(741000-800000)] with accuracy 0.5806 on (18,9,3,0,1,0,0)

IF maxsıc < 22.30 and maxsıc < 14.00 and maxsıc < 3.35 THEN sutüket in [(601000-700000)] with accuracy 0.7778 on (0,4,14,0,0,0,0)

IF maxsıc < 22.30 and maxsıc < 14.00 and maxsıc >=3.35 THEN sutüket in [(701000-740000)] with accuracy 0.5435 on (5,25,16,0,0,0,0)

5.2.5 C4.5

C4.5 is the most accurate which has accuracy = 0.6868 of the 5 algorithm for our dataset. In addition, it has also best misclassification cost = 0.2604 for 5 times execution. It has classification rules below:

For one execution:

IF maxsıc >=22.30 and rüzgar >=7.35 and rüzgar >=14.55 THEN sutüket in [(601000-700000)] with accuracy 0.3333 on (0,1,1,0,1,0,0)

IF maxsıc >=22.30 and rüzgar < 7.35 and nem < 42.35 THEN sutüket in [(801000-900000)] with accuracy 0.6667 on (1,0,0,0,2,0,0)

IF maxsıc >=22.30 and rüzgar < 7.35 and nem >=42.35 THEN sutüket in [(741000-800000)] with accuracy 1.0000 on (8,0,0,0,0,0,0)

IF maxsıc < 22.30 and maxsıc >=14.00 and yağış < 0.05 THEN sutüket in [(601000-700000)] with accuracy 1.0000 on (0,0,3,0,0,0,0)

IF maxsıc < 22.30 and maxsıc >=14.00 and yağış >=0.05 and rüzgar

>=13.65 THEN sutüket in [(701000-740000)] with accuracy 0.7143 on (1,5,1,0,0,0,0)

IF maxsıc < 22.30 and maxsıc >=14.00 and yağış >=0.05 and rüzgar < 13.65 and maxsıc < 21.20 THEN sutüket in [(741000-800000)] with accuracy 0.7619 on (16,4,0,0,1,0,0)

IF maxsıc < 22.30 and maxsıc >=14.00 and yağış >=0.05 and rüzgar < 13.65 and maxsıc >=21.20 THEN sutüket in [(601000-700000)] with accuracy 0.6667 on (1,0,2,0,0,0,0)

IF maxsıc < 22.30 and maxsıc < 14.00 and maxsıc < 3.35 THEN sutüket in [(601000-700000)] with accuracy 0.7778 on (0,4,14,0,0,0,0)

IF maxsıc < 22.30 and maxsıc < 14.00 and maxsıc >=3.35 THEN sutüket in [(701000-740000)] with accuracy 0.5435 on (5,25,16,0,0,0,0)

IF maxsıc >=22.30 and rüzgar >=7.35 and rüzgar < 14.55 and minsıc < 13.40 and rüzgar >=10.55 THEN sutüket in [(901000-1400000)] with accuracy 0.4000 on (2,0,0,0,1,2,0)

IF maxsıc >=22.30 and rüzgar >=7.35 and rüzgar < 14.55 and minsıc < 13.40 and rüzgar < 10.55 and nem < 45.85 THEN sutüket in [(801000-900000)] with accuracy 0.7778 on (0,0,1,0,7,0,1)

IF maxsıc >=22.30 and rüzgar >=7.35 and rüzgar < 14.55 and minsıc < 13.40 and rüzgar < 10.55 and nem >=45.85 THEN sutüket in [(741000-800000)] with accuracy 0.5833 on (7,0,0,0,5,0,0)

IF maxsıc >=22.30 and rüzgar >=7.35 and rüzgar < 14.55 and minsıc >=13.40 and maxsıc < 28.90 THEN sutüket in [(801000-900000)] with accuracy 0.9167 on (0,0,1,0,11,0,0)

IF maxsıc >=22.30 and rüzgar >=7.35 and rüzgar < 14.55 and minsıc

>=13.40 and maxsıc >=28.90 and rüzgar < 8.50 THEN sutüket in [(901000-1400000)] with accuracy 1.0000 on (0,0,0,0,0,5,0)

IF maxsıc >=22.30 and rüzgar >=7.35 and rüzgar < 14.55 and minsıc >=13.40 and maxsıc >=28.90 and rüzgar >=8.50 THEN sutüket in [(801000-900000)] with accuracy 0.6296 on (0,0,0,0,17,10,0)

5.3 Comparison Table

Table 2 – Efficiency Comparison between Algorithms

Algorithm	Average Accuracy	Average Cost
C4.5	% 69	0.2604
ID3	% 68	0.3527
CART	% 59	0.4668
CHAID	%53	0.4395
QUEST	%55	0.4853

CHAPTER 6

CONCLUSION

In this thesis, we applied several decision tree algorithms including Quest, ID3, C4.5, CART and CHAID on to the problem domain that is "Typical of a water distribution system of Ankara city" in order to optimize controls and other factors in the domain. However, optimal control requires an ability to precisely predict short-term water demand so that minimum cost pumping schedules can be prepared. Based on such an implementation we compared the decision tree algorithms with respect to their effectiveness in the domain. [14]

The reason of choosing this domain was optimizing control of operations in a municipal water-distribution system so that electricity costs can be reduced greatly, which is the most prominent factor in the domain as well as realizing other economic benefits. Today majority of water distribution systems still uses heuristics of experienced people to control required operations. This thesis provides a step towards an autonomous system that takes intelligent decisions to control and coordinate the operations in a typical water distribution system. With minimized human factors, effects of incorrect decisions will also be minimized.

A great effort is given for data gathering (knowledge acquisition) and pre-processing phase of data. For this purpose, data's has taken from the Government Meteorological Head office in Kalaba and İvedik Water distribution and Refinery organization.

There were many attributes in meteorological data. However, we chose the best attributes to define the appropriate dataset. These attributes are; Day of the week, maximum temperature, minimum temperature, humidity, rainfall and snowfall and average wind speed.

After collection of data, pre-processing part comes first. In the pre-processing part, one of the essential parts of data preparation is attribute categorization, an important and potentially overlooked piece of the data-mining puzzle. If attributes are improperly categorized, then the data-mining analyst will miss significant results. Categorizing attributes allows the data miner to find patterns about groups of numbers. The goal of any data miner is to find meaningful results, and properly categorizing attributes can help to do so. A standard categorization will be grouping numbers. There are five factors that an analyst may consider when categorizing data. The first factor is the number of categories. Too many categories can make the data overly specific and by doing so, the data miner may miss general patterns. A second factor is the distribution of the data set into the categories. One method of data categorization will have the data evenly distributed into the categories. A third factor is the range of the categories. If a range of a category is too big, it may contain data that should not logically be grouped together. A fourth factor is the consistency of the range of the categories. Keeping the range of the categories consistent helps keep the analysis of the attribute well organized. The last factor to consider when categorizing data is logical breaks between categories. According to these rules, we made attribute categorization for water consumption data.

The algorithms, which we were, choose, may or may not be the best algorithms in decision tree learning. However, we decided to choose them and applied on our dataset. Therefore, the results showed that C4.5 and ID3 are the most accurate and less costly algorithms that we have. Of course, other three algorithms have some advantages on different ways, someone who want to execute different parameters for different domains.

In the future, some other methods of supervised learning, neural network applications can be done on this domain. We can expand the dataset not for a city but for all cities of Turkey.

REFERENCES

[1] **Russel and Norvig**. *Artificial Intelligence: A Modern Approach, 1995.*

[2] **Leake, David.B**. *Artificial Intelligence. From Van Nostrand Scientific Encyclopedia, Ninth Editin,.IındianaUniversity* Wiley, New York, 2002.

[3] **R.S. Michalski, J.G. Carbonell, and T.M. Mitchell**. *Machine Learning: An Artificial Intelligence Approach, Vol. 1*, Morgan Kaufmann, San Francisco, 1983.

[4] **Wilson, Bill** .*The Machine Learning Dictionary for COMP9414,1998.*

[5] **Zarndt, Frederick**. *A Comprehensive Case Study: An Examination of Machine Learning and Connectionist Algorithms*, June 1995.

[6] *http://www.learningtheory.org/*

[7] *http://www.aaai.org/AITopics/html/machine.html*

[8*]* *http://aima.cs.berkeley.edu/ai.html#learning*

[9] *http://www.ics.uci.edu/~mlearn/*

[10] *http://www.cs.cmu.edu/afs/cs/project/jair/pub/volume4/kaelbling96a-html/rl-survey.html*

[11] **Mitchell, Tom**., *Machine Learning,* Mcgraw Hill, 1997.

[12] *http://www.stat.wisc.edu/□loh/quest.html*

[13] *http://eric.univ-lyon2.fr/*

[14] An**, Aijun. Chan, Christine.Shan, Ning. Cercone, Nick. Ziarko, Wojciech.** Saskatchewan, Canada. *Applying Knowledge Discovery to Predict Water-Supply Consumption*, University of Regina, 1997.

.