Adaptive Automated Diagnosis

Magnus Stensmo

November 1995

Submitted in partial fulfillment of the requirements
for the degree of *Teknologie Doktor* (Doctor of Philosophy)

Department of Computer and Systems Sciences
Kungliga Tekniska Högskolan
(Royal Institute of Technology)
Stockholm, Sweden

Doctoral Thesis
Royal Institute of Technology

ISBN 91-7153-415-6

Till Barbro och Nils-Anders

November 21, 1995; Revised February 23, 1996

This thesis is available at http://www.cnl.salk.edu/~magnus/thesis.ps.Z

ADVOKATEN: Kanske ni har lust att promoveras och få en lagerkrans?
OFFICERN: Jaa, varför inte? Det är alltid en liten distraktion...

August Strindberg

Abstract

Diagnosis of machine failure or human disease are common everyday tasks that are performed daily by many experts in different professions. Its automation would yield many advantages, mainly monetary for machine diagnosis, but also in terms of reduced suffering or pain in medicine. Yet it has proven very hard to automate diagnosis in a robust way. One of the main reasons is that an expert is often not aware of how the diagnosis is performed, so that a formal or algorithmic description can be hard to attain. This thesis presents a complete system based on probability, utility and decision theory for automated diagnosis. The system has the ability to learn from a case database, thereby overcoming problems due to manual specification. Missing data, both at the time of diagnosis and in the training set, are always present in diagnosis due to its nature; initially little is known, then questions are asked and tests made to receive more information. The joint probability distribution of the data is modeled with mixture models to give the correct result in any such situation. The parameters are determined by the Expectation-Maximization algorithm which can also handle missing data in the training set. Decision theory is used to find the most informative next question to ask, or test to do, in order to minimize the total cost for the diagnosis. It is also used to decide when to stop requesting more information. Temporal-difference reinforcement learning is used to automatically find good utility values by using an expert's assessments of diagnosis sequences created by the system as a reinforcement signal. This also allows for the use of non-linear utility functions instead of the customary linear functions. The complete system is exemplified on several small databases and on a medical data set for heart disease.

Acknowledgements

Graduate School proved to be a very large and seemingly endless labyrinth. It took me quite a while to realize that there was in fact a way out of it. On my way I found some blind alleys but also stretches of pleasurable straight paths. It had not been possible to find the right way out without a lot of help.

My first thanks goes to Terry Sejnowski who has supported and sponsored me in many ways to really make it happen. He allowed me time to back-track and find my own way, with incisive comments, ideas and encouragement along the way. I might not have found the goal in this millennium without him.

Terry's Computational Neurobiology Lab at the Salk Institute in La Jolla has been my home away from home for the past three plus years. Time in the lab has always been educational and interesting. Life in the CNL "family," with all those afternoon teas, has been very pleasant. And who can fail with such a great library of journals, books and people?

On the home front, Carl Gustaf Jansson guided me through the final intricate twists and turns, and out of the labyrinth. Without his skill, persistence and comments, I would have been there much longer. Eva Jansson made sure that all paperwork (including this thesis) ended up where it should.

This thesis manuscript was read by several people whose comments have made it much better. First I must thank Ann Frisinger who was by far the fastest reader. The thesis committee and my opponent, Pekka Orponen, have provided many insightful comments. Iris Ginzburg, Terry Sejnowski and Mats Danielson also provided comments which they are thanked for.

The work behind this thesis has benefitted from many comments and discussions with Peter Dayan. During the past year I have had interesting discussions with Iris Ginzburg. Rich Zemel, Nici Schraudolph, Olivier Coenen and Paul Viola have read and commented on some papers this thesis is based on.

Administration and all those other things at CNL have been handled superbly by Rosemary Miller, Hillary Chase-Benedetti, Alicia Vana and Leslie Chaden. Computers have been maintained equally superbly by Dave Lawrence, Bryan Nielsen, Tom Bartol and Tim Leek.

Current and past office and lab mates—too many to name—are thanked for numerous discussions and fun. I spent about two years in the "Tower of Babbel," which at times gave full merit to its name. The last year was spent in our new quarters in what is probably Salk's most photographed office[1]. The view from the tower was much better though.

A scholarship from the Sweden-America Foundation made my coming to CNL possible which I am very grateful for. The heart disease database that was used in this thesis comes from the UC Irvine Repository of Machine Learning Databases [Murphy and Aha] and originates from R. Detrano, The Cleveland Clinic Foundation.

Thinking back when I started my graduate studies I could hardly imagine that I should spend the majority of the time with Terry Sejnowski in California. Had it not been because of some unexpected events early on, I would probably not have ended up at this place and time—so I guess Pangloss was right after all. I must now cultivate my garden.[2]

[1] http://dirac.salk.edu/cgi-bin/diracam

[2] If you really don't want to read any further there is a summary on page 106.

Contents

Chapter 1

Introduction

1.1 Diagnosis

Diagnosis is the process of identifying a disease or disorder of a patient or a machine by considering its history, symptoms and other signs. This is done by examination in various ways followed by reasoning to find the most likely disorder. It is a common and important problem that is performed daily by many people in different professions. Some examples are physicians who diagnose diseases, and mechanics who find what is wrong with your car. The importance of a quick, consistent and reliable diagnosis is immediately evident. This thesis aims to automate this process.

An essential part of the diagnosis is the reasoning. False diagnosis usually result from wrong interpretations of findings or the wrong conclusions from the observations. Here is what one encyclopedia writes about diagnosis:

> While the task of collecting data on the patient-history, physical examination, special examinations, and laboratory tests—requires accurate observation, the process of correct reasoning is, in the last analysis, more important. False diagnoses often result not from false data but from faulty interpretation. To assemble the data, to make use of the relevant and to discard the irrelevant information, and to pass final judgment—all of which are important steps in making the diagnosis—often requires a mental ability of the highest order.
>
> *Diagnosis* [Encyclopædia Britannica, 1992]

This is certainly true. The training time for human experts are, as we all know, often quite long as are the requirements for reasoning and judgement. Many times only knowledge through experience can solve a problem. This is the case when a domain is not known and understood in enough detail. Many domains are not easy to analyze or formalize theoretically which means that an exact model or algorithm cannot be easily formulated.

Being this complicated, diagnosis has been called an art form. Here is one example of that, in reference to medicine:

> The formulation of a differential diagnosis[1] is one of the most important

[1] In medicine, *differential diagnosis* is the name of the process to find which out of several diseases a patient has.

> and intellectually challenging aspects of medical reasoning. When a clinician encounters a patient, the clinician faces a vast amount of information: the patient's lifelong personal and medical history, the patient's report of the current medical problem; and the results of numerous examinations, procedures, and tests. In addition to this information the clinician must have a tremendous amount of knowledge about health and disease. Somehow, seasoned clinicians are able to sort their way through the details, clear the confusion, and make the diagnosis.
>
> *The Art of Diagnosis* [Eddy and Clanton, 1982], p. 1263

Contrary to this opinion, I do not believe it is an art form. An art is something we don't fully understand. We can, however, understand, formalize and automate diagnosis successfully, as this thesis will demonstrate.

Diagnosis has proven hard to automate and formalize because experts often do not know exactly how they solve a problem. They do it, but can't always explain how or why, often not even after the fact. Their skills are acquired through both study and experience. The objects they examine are often complicated and not fully understood, as in the case of the human body, but even a man-made machine can behave in unexpected and complicated ways if it is very complex. For these reasons, a procedural description may be hard or even impossible to attain. It is not an easy task to write a program that performs diagnosis at human levels.

Tools that help non-experts to diagnose, or that simplifies diagnosis for experts are much in need. For industrial machines it is usually because of monetary reasons. Down-time is very expensive, and when a machine is out of order it can disrupt other things down the line. Preventive maintenance may help avoid complicated problems. For medicine it is for monetary reasons too, but perhaps more importantly we would like to reduce pain and suffering for the patient.

There is also a resource allocation problem. In the Western world there are plenty of physicians, but not so everywhere else. For remote areas a computerized tool could be useful to, e.g., help a physician diagnose something outside of his immediate area of expertise.

The type of diagnosis discussed in this thesis is sometimes referred to as *maintenance diagnosis*. There is also another kind, called *operative diagnosis*, which applies to the case when you cannot add sensors or run tests that will render the object inoperable. A typical situation could be a space craft that has to be on-line all the time. Diagnosis is then part of a larger problem known as Fault Detection, Identification and Reconfiguration (FDIR), that includes all steps required to render a faulty device functional again, perhaps with a limited set of available resources. This is discussed further in [Crow and Rushby, 1994].

Diagnosis is a process. There are a number of clearly defined steps—although different for each object—that are performed. At each step new information is acquired, incorporated into the knowledge base, and used to refine the result. Starting from a few prior observations the diagnosis is vague. Then, as more information is gathered, the diagnosis becomes more precise. An efficient way to acquire new information dynamically is needed that does not waste resources on unnecessary examinations.

A note on the title. *Automated diagnosis* means that the system will suggest questions but not answer them, which then would be called automatic diagnosis. This is discussed in the next section. *Adaptive* means that the system learns from data and past usage. This is introduced in Section 1.3 on page 8.

1.2 Automated Diagnosis

1.2.1 Diagnosis as classification

Diagnosis can be viewed as classification. This is one of the main assumptions in this thesis. The goal is to classify what disorder or disorders are present given current observations. The observations and examinations lead the expert to a conclusion or diagnosis of what is being observed. It can for example be determining a disease from observed symptoms or an engine fault from the (mis)behaviour of an engine. If viewed as a classification problem we can study the problem using classification techniques and can consequently use classification of any data for test examples.

In traditional systems, information, whether as rules, text book knowledge or experience, has to be made explicit for its use. This thesis takes a different approach and assumes that the information, whether explicit or hidden, about a specific problem exists in a set of cases for the domain. These cases are previous occurrences of the disorders that have been recorded. They can for example be acquired from patient records or shop maintenance records. Sometimes a set of cases is very similar—apart from small and unimportant random variations, *i.e.*, noise—and become prototypes or syndromes. It is assumed that there really is something wrong with the device that is diagnosed.

The underlying relationships between the disorders and observations may be obscure, but they are nevertheless present provided the database is sufficiently complete. They may of course be obscured within a large set of seemingly irrelevant or unrelated cases. There is an underlying assumption in this work that malfunction does not occur completely at random, in which case there would be no such statistical correlations. This could be the case if the database consisted of totally irrelevant variables, but why anybody would record such uninformative "cases" eludes me. If not enough meaningful variables are recorded, the database is not useful.

By considering diagnosis as a statistical inference problem we can attempt to understand the underlying relationships. Machine learning using previous observations as a knowledge base makes it possible to extract and use this information without the need for an expert with long training that tells us what to do.

The kind of databases needed are not very common now. The ones that are available now are often proprietary due to the large effort that is required for their collection. In the future databases should be much easier to obtain. For example, there are already hospitals that have completely abandoned paper patient records for computerized ones. Substantial preprocessing is needed but the data will at least be available in a machine readable format. Machine faults can also be created artificially in the laboratory by systematically breaking components and observing the results.

Since diagnosis is viewed as a classification problem it can be studied with any kind of database. When the words *disease* and *symptom* are mentioned below, *disorder* and *observation* can be equivalently substituted, as can *outcome* and *evidence*. The domain independence that results from the use of machine learning approach is one of the important features that makes this approach appealing.

Only an expert can understand a medical database, while anybody can understand a database with, *e.g.*, animals. This has been beneficial during exploration and development of the systems presented in this thesis. Some of these small test databases are presented in Section A.2 (page 114). For the more complete systems in the later chapters a real-world medical database was used, see Section A.1 on page 114.

1.2.2 Probability estimation

Diagnosis is a probability estimation problem. The goal is to find the probability distribution—a number of probabilities—over the disorders. They signify the strength, or degree of belief, in the result. The probabilities are, more precisely, conditional probabilities that are conditioned on what has been observed. There is a set of possible disorders and a set of variables that describe the situation in the form of symptoms observed and results of laboratory tests. In a medical context the disorders are diseases and the variables are any data about the patient that is needed for the diagnosis.

The diagnosis is clear and reliable when one or a few of all of the possible outcomes are differentiated from the others, making its probability come close to one. If on the other hand the diagnosis is inconclusive, so that several of the outcomes are more or less equally likely, or none of the outcomes reach a level that we are satisfied with, then additional information is needed to make it more conclusive and reliable.

Another important aspect is that, from the nature of the task, there are always missing pieces of data when there is diagnosis. There are variables we do not know the value of and tests we have not done. Initially when an object is examined there are a few clues, but the rest of the variables are essentially unknown, except perhaps for prior probabilities for the occurrences of the outcomes. Additional information is then actively sought.

These missing pieces of data are often just ignored, but they may significantly influence the result. In other applications missing data can perhaps sometimes be ignored if they are rare. For diagnosis it will be important to be able to handle it in a principled way since there will always be missing data due to the nature of the process. Probability theory provides for a principled handling of this.

1.2.3 Asking questions

One way of operation of an automated classifying diagnosis system could be to simply require the presence of all possible evidence at once, and then from that deduce the most likely diagnosis. This is admissible and sufficient from a classification point of view and has been pursued to quite a large extent (Section 6.3.1 (page 101)).

There are, however, some drawbacks with that approach. There are often costs associated with observations. For example, a laboratory test may have to be done, or we may have to disassemble a machine. These things both takes time and will cost a lot of money. For a patient a procedure may be painful, critical to the health, or just unpleasant. We therefore only want to find out about those variables that are necessary to reach to a sufficiently good result.

If the classification system instead can ask questions to the user only the evidence necessary to reach to a conclusion need to be given, provided that appropriate questions are asked. The system should choose questions based on its current knowledge and can use a cost associated with an attribute when choosing them. By necessity, there will always be substantial amounts of missing data as discussed above. For this to work we need an accurate estimate after each step.

The total cost should be minimized in all diagnosis. This will lead to an efficient and reliable diagnosis system. Cost is measured differently depending from whose point of view it is seen. The patient is worried about his pain or future or present disability, the insurance company is worried about how much money is spent. Some clinics or doctors may perhaps want to increase their earnings by performing unnecessary tests.

Probability theory is a good description for uncertainties and in the handling of missing data, but when there is a need for decision making something more is needed. We find this in utility theory, which can value the usefulness of different states so that they can be compared, and in decision theory, which is a combination of probability and utility theory. These methods will help us find the best questions to ask and in what order. These matters are discussed in Chapter 2 on page 24.

1.3 Artificial Intelligence and Neural Networks

1.3.1 Learning machines

This thesis falls within the subfields of *Neural Networks* (NN) and *Artificial Intelligence* (AI) of Computer Science and related fields. The goals for both of them are in fact the same as we will see later, so they are quite similar, but there are also major differences. The differences lie mainly in the history of the fields, and in their traditional approaches to solutions to the problems.

Historically Artificial Intelligence has been *symbolic* and Neural Networks has been *distributed* in knowledge representation and reasoning. This means that in AI a concept would be represented by one object or symbol, e.g., a word. In Neural Networks, the representation would be distributed over several objects that are shared by related concepts.

If a human is to specify something, the symbolic representation seems quite natural. The inspiration for this probably comes from how we perceive ourselves reason and think through introspection. Logic was created to describe and mechanize human reasoning and thinking dating back to Aristotle's syllogisms. Unfortunately logic covers only parts of human reasoning, as has been found when it has been used as a representation language. Extensions to traditional propositional or first-order logics have in general not been too successful.

If the symbols or concepts are known, logic can be useful, but problems arise when a suitable set of symbols is not known. What should they then be? There is often no answer. The symbolic approach in this way goes from high-level concepts down to the data. This is called *top-down*.

In the Neural Networks field, on the other hand, with distributed representation, it is thought that one particular concept is spread out over several subparts in a network, so that related concepts have parts in common. It is then thought that—at least in theory—new concepts can be formed by novel combinations. Inspiration comes, as suggested by the name, from how the brain work, or at least how we think it works.[2] It is thought that many neurons represent concepts in a distributed fashion, but exactly how this is done is not known.

How networks of units are modeled is described below. It is possible to design learning algorithms that can learn parameters of such systems so that they can perform desired tasks. Training examples for this learning come from a database of examples in the same way that we can learn. This approach is therefore going from the data to the final concepts, called *bottom-up*. Many learning algorithms can additionally be derived from probability theory which is a more general representation for knowledge than logic.

Interestingly enough, these fields sometimes get combined, so that there are, *e.g.*, learning methods in AI, or symbols in Neural Networks. Machine Learning is in fact now the largest subfield of Artificial Intelligence. What is the most suitable representation and modeling tool depends on the problem that we would like to solve.

There is sometimes a distinction between *shallow* and *deep* for reasoning and representation in a system. The former means that there is no background knowledge built into the system. The system then knows only what is explicitly provided for the solution to the problem. Deep structure means that background knowledge is there too. Every one of us, whether we want it or not, always have plenty of background knowledge that we have acquired

[2] Somebody has said that "the brain works the way you think."

over time. A computer does not have any.

Background knowledge can provide useful additional information but also a bias that may not always be beneficial to us. It should at least be known. It is not easy to come up with a representation that can be useful in different systems for different domains, but work is in progress in the Cyc project [Guha and Lenat, 1990]. The brain is very good at doing this too.

1.3.2 Goals

What are the goals of Artificial Intelligence and Neural Networks? There are in my mind two major goals. Not everybody will subscribe to the same ones, but I think most would agree with the following.

To understand how the brain works. After all, it is probably both the least understood and most complicated object on earth. Everybody has a nervous system and can use it for an impressive multitude of tasks almost effortlessly and instantly. Understanding how this impressive machine works would be a very worthwhile goal in itself.

To build better computers. Today's computers (or machines—the computers are just our most complex ones) are very good at some tasks, but are all the same hopelessly bad at other ones. They can add billions of numbers each second, but they cannot reliably recognize a face in a crowd or understand language. We would like future computers to have these abilities and more.

These goals clearly are not attainable right now, but they do not seem impossible. Some believe that it is not to be possible to achieve them at all and there is a never-ending philosophical discussion on the subject. The following quotation gives some insight into this fascinating problem. For those who believe that reasoning can't be automated, the brain provides an existence proof.

> AI addresses one of the ultimate puzzles. How is it possible for a slow, tiny brain, whether biological or electronic, to perceive, understand, predict, and manipulate a world far larger and more complicated than

> itself? How do we go about making something with those properties? These are hard questions, but unlike the search for faster-than-light travel or an anti-gravity device, the researcher in AI has solid evidence that the quest is possible. All the researcher has to do is look in the mirror to see an example of an intelligent system.
>
> [Russell and Norvig, 1995], p. 3

1.4 Neural Network Models

1.4.1 Motivation

Ideas and inspiration for artificial neural networks comes from neuroscience. This is due to the fact that man can do many tasks better than computers although our computing elements (the nerve cells or neurons) are on the order of one million times slower and much more unreliable than those of a computer. The nerve cells are not very precise or capable of stable output for very long times. It is not the speed or accuracy of individual elements that does the job, it is a complicated interaction and cooperation of thousands and millions of them.

The brain can be viewed as a massively parallel machine with an estimated 10^{10}–10^{11} nerve cells.[3] Each cell has on the average about 10^{3}–10^{6} connections to other cells. Synapses connect the cells together in giant networks and mediate transmission of signals. When learning occurs this is where changes happen. It is now known that the efficacy of the synapses and the amount of them change with learning.

The brain is—as we all know— not very good at basic arithmetic or at keeping track of details. It is instead good at recognizing things, drawing conclusions, finding analogies, *etc.*— something we call pattern recognition. If a computer is to search a database and we give it more details about what we are looking for, then the search typically will take longer time to finish than without these extra details. A human doing the same thing will

[3] To better grasp the magnitude of this huge number it is interesting to note that there are about this amount of stars in our galaxy.

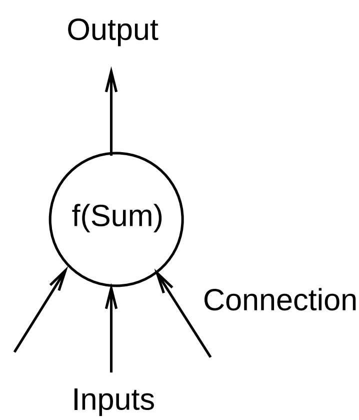

Figure 1.1: A unit in an artificial neural network.
A unit computes a weighted sum of its inputs and passes the result through a usually sigmoid non-linearity f to produce its output, which in turn is the input to other units.

find matches faster with more information. This simple example shows that there is a fundamental difference in how these two systems function.

Apart from trying to understand how the brain functions, it is interesting to try to automate what we are good at, be it pattern recognition (like speech or vision) or, *e.g.*, diagnosis. The next section describes some of the artificial models that have been used for this purpose.

1.4.2 Artificial models

Artificial Neural Networks compute and process information by transforming input to output by passing signals through units that are connected to each other. The units process and combine its inputs and emit an output signal. The connections between units have different strengths or weights so that not all units are equally important in the combination. If the connection strength is zero there is no connection at all between two units.

A typical unit (Figure 1.1) computes a weighted sum of its inputs. This sum is then passed through a non-linearity on its way to the units that are connected to it. The units perform a non-linear transformation of this

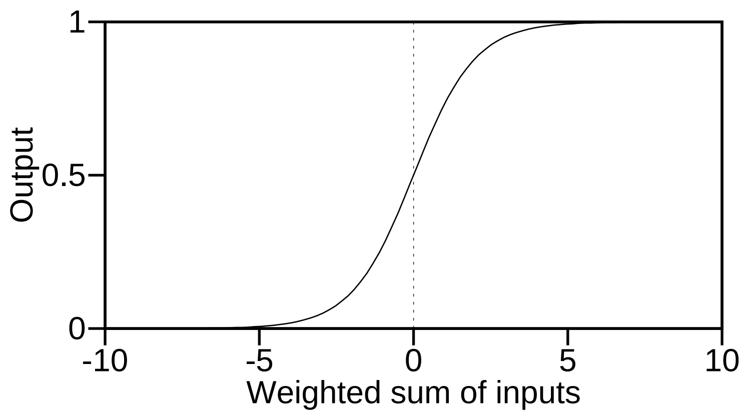

Figure 1.2: A sigmoid output function.
This sigmoid function has the form $f(x) = 1/(1+e^{-x})$, where x is the input. The input is monotonically transformed into the fixed interval [0,1].

weighted sum by applying an output or squashing function. It is often a sigmoid, *e.g.*,

$$f(x) = \frac{1}{1+e^{-x}}.$$

The sigmoid is monotonic and has a simple derivative which is used by the learning algorithms. The derivative with respect to the input sum x is

$$\frac{\partial f}{\partial x} = \frac{e^{-x}}{[1+e^{-x}]^2},$$

or simply $f(1-f)$. The output function limits the range of the signal as seen in Figure 1.2.

Units are put together in layers. Each unit in one layer sends its output to all units in the next higher layer. This is called a *feed-forward network*. A *recurrent* system has connections that also go in the other direction. If the system has one layer of connections between the layer of input units and the layer of output units it is called a *one-layer* system (Figure 1.3 (over)). If it has additional units that are neither input nor output, called *hidden units*, it is a *multi-layer* system (Figure 1.4 on the following page). In a *fully-connected* system all units in one layer are connected to all units in the next layer.

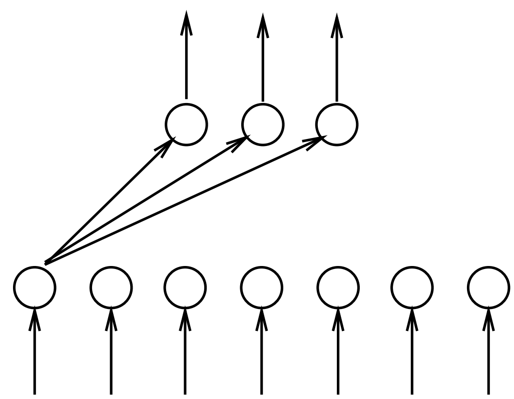

Figure 1.3: A single-layer neural network.
The circles are the units or nodes in the system. Arrows indicate connections. Note that connections are only shown for the first unit to the units in the second layer, all other units have similar connections.

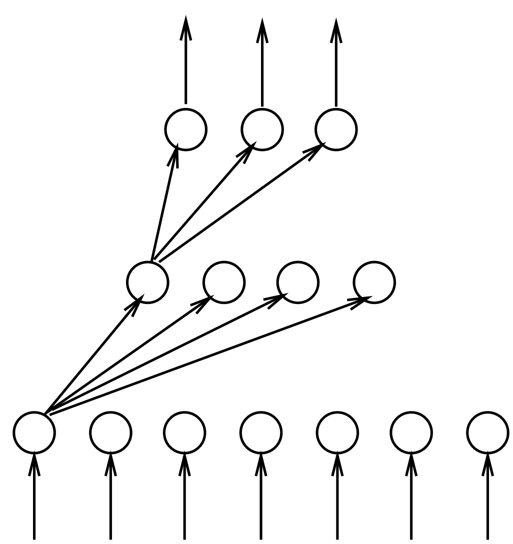

Figure 1.4: A multi-layer neural network.
There are one or more hidden layers between the input and output layers in a multi-layer system. Note that connections are only shown for the first unit in each layer.

One-layer networks

One-layer systems, Figure 1.3, are quite straight-forward to analyze and use. There exist learning algorithms that can be proven to converge, which is not the case for multi-layer systems. For one-layer systems, the patterns can only be learned if they are not too similar, *i.e.*, not overlapping, otherwise cross-talk between units will degrade the result.

When it is not possible to learn two patterns, the system learns a pattern that is somewhere in between the two. The network considers the two patterns to be examples of the resultant pattern corrupted by noise and generalizes to this "prototype" pattern. The generalization capability will benefit us if we have patterns that are corrupted by noise, otherwise over-generalization and cross-talk will destroy the result.

These two features unfortunately come together, recovery from noise and generalization. What is noise and what is another pattern is the observer's interpretation—the system cannot know anything about it since its knowledge is shallow.

One-layer systems can be viewed as a weight matrix that is multiplied by the inputs. Being a linear system, the matrix elements can also be learned by methods from linear algebra.

Multi-layer networks

It can be shown that in a one-layer system, the input patterns have to be linearly independent or at least statistically independent to avoid corruption by over-generalization. If this is not the case we can introduce internal transformations through hidden units. Several hidden layers will do several transformations in sequence. This leads to very powerful and general architectures called multi-layer systems (Figure 1.4).

There is however a drawback with this, a "credit assignment" problem of how to set the weights for the hidden units in multi-layer systems when all input units and all previous layers can affect the result. The learning algorithms for hidden unit systems, *e.g.*, back-propagation solves this but do not always converge, and there is also no way of knowing what architecture to use for a network, *i.e.*, the number of hidden units, layers and connections.

Multi-layer systems do the mapping mentioned above in stages. It is important that there are non-linearities since otherwise these stages of a multi-layer system would reduce to a one-layer system (since the product of two matrices is a matrix) with its restrictions.

To be able to draw conclusions from these models, simulations are necessary, except perhaps for very simple systems, because of the non-linearities of the units and to the lack of powerful analytical methods for non-linear systems.

Learning

Learning in a neural network adjusts the connections between the units so that an input gives a desired output or fulfills some other criteria. There are three different kinds of learning.

Supervised learning adjust the connections so that a specified input will match a given target for the output. Labeled examples are required. *Unsupervised learning* use unlabeled inputs and groups them together in clusters. There is also a third kind called *reinforcement learning* where a teacher signal is only given at some times during the learning, instead of at every presentation as in the supervised case.

Gradient descent is a simple form of learning. Parameters are adjusted so that a measure of the output error compared to the target values is minimized. Often a sum of squared errors measure is used. A correction term is calculated for each connection weight.

If we think of the error measure as an energy landscape we move in the gradient direction to the minimum which is the optimal point for the whole system. In the single-layer case this is easy since there is only one minimum. In the multi-layer case the error information has to be moved backwards through the layers. This is accomplished by the chain rule for derivatives in a process called *back-propagation* of error signals [Rumelhart *et al.*, 1986b]. There can also be many local minimum points that we can get stuck in on our way to the global minimum.

Global and Local models

Each layer can be viewed as doing a transformation of an input pattern or a vector of input values. A neural network maps vectors in the input space to vectors in the output space. The non-linearity is crucial in the multi-layer system since without it the layers would collapse to one, as discussed above. These systems are *global* in the sense that all of the inputs are connected to all of the units in the next layer. This is the kind of model that is used in Chapter 3 (page 42).

A *local* system has a different design philosophy. The idea is to place local so called *basis functions* over the data points. These are then combined with the data to form the output. Often Gaussian functions are used since they have nice properties. These local functions are not completely local, but the values some distance away, say more than three standard deviations, from the center do not contribute very much to its output.

It can be shown that a linear output layer is sufficient if Gaussian basis function are used. The resulting systems are called *Radial Basis Function* (RBF) networks [Broomhead and Lowe, 1988; Moody and Darken, 1988; Moody, 1989] if the basis functions are radially symmetric Gaussians. These system have the same capabilities as global models [Park and Sandberg, 1991]. Parallel to these developments there are models in statistics called *Mixture models* that result in similar models. In Chapter 4 on page 60 these models are discussed in detail.

Artificial neural networks also deal with other topics, *e.g.*, self-organizing maps and more realistic neural models of biological neural networks. These are not addressed in this thesis since such techniques have not been used for the solution of our problem.

[Haykin, 1994], [Hertz *et al.*, 1991] and [Rumelhart *et al.*, 1986c] are good books about artificial neural networks. [Anderson and Rosenfeld, 1988] contains a collection of many important and classical papers with introductions by the editors. Machine Learning and Artificial Intelligence is well covered in [Russell and Norvig, 1995].

1.4.3 Historical overview

Interest in the subject of artificial neural systems has peaked a couple of times in the past half century, and some influential references are given here.

In the 1940's McCulloch and Pitts connected neural networks with logical calculus [McCulloch and Pitts, 1943] so that neurons were supposed to be or- and and- gates in circuit diagrams. Donald O. Hebb made a famous conjecture about how learning will change the synaptic strengths [Hebb, 1949], which gave rise to the term *Hebbian Learning*.

In the 1960's Frank Rosenblatt presented the influential *perceptron* [Rosenblatt, 1962] that gave rise to intense activity in that decade. It was more or less punctured by the appearance of the negative and quite pessimistic book *Perceptrons* by Marvin Minsky and Seymour Papert [Minsky and Papert, 1969], who claimed perceptrons were marginally useful if at all, after which much of the interest in the field was lost.

The 1970's were the dark ages and not until in the 1980's did interest peek again. It was inspired, perhaps most importantly, by John Hopfield, who drew analogies from error surfaces to energies in physical systems [Hopfield, 1982], and the discovery (by several independent groups) of a learning algorithm for multi-layer networks called *back-propagation*, see, e.g., [Rumelhart *et al.*, 1986b]. Another very interesting idea was also based on physics, the *Boltzmann machine* [Ackley *et al.*, 1985].

After this interest has continued to grow. Many of these above-mentioned classical papers can be found in [Anderson and Rosenfeld, 1988].

Interest in adaptive control and learning and adaptation in biological systems have given rise to *reinforcement learning* where learning can take place even if feedback is only available at certain points in time, as opposed to always, which is required for the methods above [Barto *et al.*, 1983; Sutton, 1988].

More recently new learning methods have been invented, perhaps the most interesting called the *Helmholtz machine* [Hinton *et al.*, 1995], which is a stochastic model that can handle probabilities and create hierarchical internal representations.

1.5 Thesis Overview

1.5.1 Chapter by chapter

The rest of this thesis is organized as follows.

In Chapter 2 (page 24) a background on probability, utility and decision theory is presented. It is important to understand that this work is based on probability theory and that in general a complete model of this will become intractable unless some simplifying assumptions are introduced. Different assumptions will not result in equally good models, as is demonstrated in the next three chapters. Utility and decision theory is necessary to be able to handle decision making—in this case the selection of the most informative question to ask, and to decide when to stop asking questions—in a principled way.

My first attempt to build an automated diagnosis system is presented in Chapter 3 on page 42. It is based on a technique called Simple Bayes, where everything is assumed to be independent and unrelated to every other thing in the model. This can work for some simple problems, but fails for more complicated ones. A good analogy is with linear systems that are good for some things, but fall short in other instances. The question selection here, which is of high importance for diagnosis, is based on a heuristic. This work was done 1989-1991.

A significantly improved probabilistic model is presented in Chapter 4 (page 60). This is a more general model that assumes much less about the underlying probability distributions. Ideas from decision theory are also incorporated so that question selection also can be performed in a principled way. The utility model, with which different states of the system are evaluated for the question selection, is linear. A simple approximation for utilities is used. This work was done 1994-1995.

The utility model is improved in Chapter 5 on page 80 where a way to learn and adapt a better and more realistic utility model is presented.

Reinforcement learning is used to modify the fixed utilities that were used in the previous chapter. This will improve the accuracy and robustness of the system. This work was done in 1995.

Chapter 6 (page 92) presents alternative and mostly older approaches to automated diagnosis for comparison with the presented systems. What I perceive as problems with traditional approaches have motivated this thesis. Most of them are not adaptive and most of them cannot handle missing data.

Finally in Chapter 7 on page 106 the whole set of ideas is summarized. Some directions that seem relevant and interesting to pursue in the future are presented in order to extend and improve the functionality of an automated diagnosis system.

1.5.2 How the chapters fit together

The historical evolution of the present work is shown in Figure 1.1 with time from left to right. It also shows what is new in each chapter and what papers this thesis is based on. On the far right future work is outlined so that it is easy to see how it fits in and extends the work presented.

<table>
<tr><th rowspan="2"></th><th colspan="3">This thesis</th><th>Future work</th></tr>
<tr><th>Chapter 3</th><th>Chapter 4</th><th>Chapter 5</th><th>Chapter 7</th></tr>
<tr><td>Probability model</td><td>Simple Bayes</td><td colspan="2">Mixture model</td><td>Bayesian networks</td></tr>
<tr><td rowspan="2">Utility model</td><td rowspan="2"></td><td colspan="2">Linear</td><td>Non-linear</td></tr>
<tr><td>Fixed 0/1 approximation</td><td colspan="2">Reinforcement learning</td></tr>
<tr><td>Decision model</td><td>Heuristic</td><td colspan="3">Maximum Expected Utility (MEU) principle</td></tr>
<tr><td>References</td><td>[Stensmo et al., 1989; Stensmo et al., 1991; Stensmo, 1991]</td><td>[Stensmo and Sejnowski, 1994; Stensmo and Sejnowski, 1995a]</td><td>[Stensmo and Sejnowski, 1995b]</td><td></td></tr>
</table>

Table 1.1: Overview of the thesis.

Chapter 2

Probability and Decision Theory

This chapter gives a background on probability, utility and decision theory, all three of which are important theoretical underpinnings for the systems described in this thesis. Skip the beginning of this chapter if you are familiar with probability theory.

2.1 Probability Theory

2.1.1 Dealing with uncertainty

Probability theory is a general form of representation for different types of uncertainty. Consider the following situations:

- When dealing with a complex system, we will almost always have incomplete models to work with, either because the object that is modeled is too complex, or because a full model would be intractable. This means that when we use the model it will not always necessarily give us the correct answer. It may give several answers with different strengths, or *degrees of belief*. The uncertainty is in this inexactness. We can call this an incomplete model due to laziness, even though we may have been forced to this by reasons of tractability.

- Another cause of uncertainty may be that we don't know in detail how the system works. An example of this is an organ in the human body. Alternatively, it could be about a domain that we can't have first-hand knowledge of, e.g., another planet. We may know a lot of things about it, but not with complete certainty.

- Apart from these two causes of uncertainty there is also the important case when we are uncertain about a test result or the presence of an observation. There may be sensor noise that the system should take care of.

To solve the above problems, probabilistic representations and inference have been found natural and straight-forward. Probabilities summarize and formalize the handling of uncertainties and degrees of belief. We can find the values of some probabilities if we know others, thereby doing a kind of inference.

Diagnosis (as well as many other real world problems) is concerned with uncertainties. Probability theory is therefore a good model.

2.1.2 Probability distributions

A *random variable* is the basic entity in probability theory. It can take any of the values from its domain, which is either discrete or continuous. A discrete random variable can take values from a set. For, *e.g.*, a dice it will be $\{1, \ldots, 6\}$. A continuous random variable has a value from within its range.

Since we don't know what value the random variable has at a particular instance, we talk about how it can be distributed over its possible values. The characteristics of the random variable (e.g., the mean or the expected value) can be derived from this distribution.

To state the degree of belief that a random variable X has a certain value x, we write $p(X = x)$. The value is one when there is absolute certainty; the event unconditionally happened or will happen. It will similarly be zero if it is impossible. Anything in between means that it may have happened or may happen in the future, with various degrees of belief. $p(X = x)$ is often abbreviated $p(x)$ when it is obvious what X refers to.

$p(x, y)$, sometimes written $p(x \wedge y)$, similarly stands for the probability that x and y occurred together. It is the *joint probability* of x and y.

To complete the formal theory there are certain requirements for probabilities. For a discrete variable $X = \{x_1, \ldots, x_N\}$ it is necessary that:

1. The probabilities sum to one: $\sum_{i=1}^{N} p(x_i) = 1$.
2. The probabilities are zero or positive: $\forall_{x_i \in X} \ \ p(x_i) \geq 0$.

For the continuous case the sum becomes an integral.

Let us now say that there is a set of mutually exclusive disorders that you are diagnosing. When you're perfectly certain that it is exactly one of them the probability for that one will be one and zero for the others. In

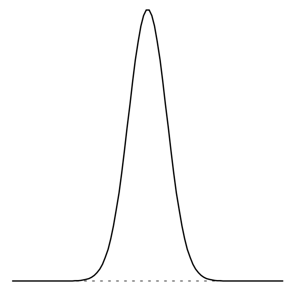

Figure 2.1: A Gaussian probability distribution.

other cases there is uncertainty and there will be a number of the other ones that are not zero. There is thus a *distribution* of possibilities.

For many naturally occurring entities a Gaussian distribution is a good descriptor. It can be demonstrated that this is the distribution that results by adding a very large number independent events together, *e.g.*, how tall people are in a city. The distribution has the familiar clock-curve shape as shown in Figure 2.1.

The following books provide background on probability theory and statistics: [Papoulis, 1984; Grimmett and Stirzaker, 1992; Allen, 1978].

2.1.3 Conditional probabilities

Diagnosis is a probability estimation problem. We seek the probability that a certain disease or disorder is present. The probabilities of the disorders are dependent or conditioned on what has currently been observed. This is called a *conditional probability*, and is written

$$p(c_i|\boldsymbol{x})$$

for the probability of disorder c_i, from a set $\boldsymbol{c}$, given a set of variables $\boldsymbol{x}$.

The definition of conditional probability is:

$$p(c_i|\boldsymbol{x}) = \frac{p(c_i, \boldsymbol{x})}{p(\boldsymbol{x})}, \tag{2.1}$$

provided that $p(\boldsymbol{x})$ is never zero.

2.1.4 Computing conditional probabilities

There are several ways to compute conditional probabilities, each with its limitations and benefits. Three of them are presented here.

Direct. The direct approach is to learn a functional approximation from the inputs, $\boldsymbol{x}$, to the outputs, $\boldsymbol{c}$. These are the conditional probabilities $p(c_i|\boldsymbol{x})$ for each disorder c_i. A neural network can do this if it has a suitable configuration (Section 6.3.1 (page 101)).

Indirect. The indirect way is to rewrite the conditional probability using *Bayes' rule*[1]

$$p(c_i|\boldsymbol{x}) = p(c_i)\frac{p(\boldsymbol{x}|c_i)}{p(\boldsymbol{x})}, \tag{2.2}$$

and estimate the parts separately by, *e.g.*, neural networks as in the direct case. The indirect method is used in Chapter 3 on page 42.

Joint probability. A third way is to model the joint probability $p(\boldsymbol{c}, \boldsymbol{x})$. If this is known it is easy to marginalize to get any conditional probability, as explained below. It is necessary to model the joint probability to be able to handle missing data in a principled way. This is the approach that is used in Chapter 4 (page 60). Why and how is explained in the next section.

See also [Hush and Horne, 1993] for a related discussion.

2.1.5 The joint probability density

In probability theory, the joint probability distribution is one of the most important of the fundamental concepts. It specifies the probabilities for each possible combination of variables that describe a problem, and has been found to be a general description for a wide range of problems. Joint here stands for the total combination of the probability variables. When

[1] This rule should really be called *Bayes's rule* since it refers to work by Rev. Thomas Bayes [Bayes, 1763], but is almost never written like that.

the joint probability distribution is known, any probability relating to the domain can be found using the following method:

Let us assume that we have the joint probability distribution of a two-dimensional discrete random distribution $p(X, Y)$. We are interested in the distribution over X, $p(X)$. It is simply found by summing over the other dimension so that the probability becomes

$$p(X) = \sum_{y \epsilon Y} p(X, y). \tag{2.3}$$

This is called *marginalization*. It is the projection of $p(X, Y)$ onto the X dimension in this case. The formula generalizes to any number of dimensions. We can in this way use (2.3) to calculate any probability or conditional probability as in (2.1) (page 26).

The joint probability density is unfortunately so general that its description and even evaluation many times is computationally intractable. For example, for a discrete problem with n two-valued (binary) variables, there are 2^n different combinations of subsets, each of which has to be assigned an estimate. The complexity thus grows exponentially in the number of variables. The implication of this is intractability for any non-trivial problem size.

We can think of all of these probabilities as entries in an imaginary very large look-up table. There will be many zero entries implying that they can be ignored since a zero entry means independence. The problem is to find *which* of these entries we can discard without dire consequences.

It is necessary to have access to the joint probability distribution to be able to solve the missing data problem in a principled way [Ahmad and Tresp, 1993], as discussed in the next section.

2.1.6 Missing data

As discussed above, missing data are abundant in diagnosis and must be handled in a principled manner. There are two types of missing data that we have to be aware of:

Missing data at the time of diagnosis. When the patient comes to see a doctor often not much more than what is immediately evident is known, *e.g.*, the sex and age of the patient. To find out more the patient is interviewed and some laboratory tests are perhaps performed. All of the unknown variables are missing in this terminology.

Missing data in the case database. The learning techniques have to be able to handle missing values in the case database. This is very common in databases for real world problems. Perhaps there was no opportunity to register all variables, perhaps several databases with slightly different contents were merged, there can be many reasons.

The next sections looks into these two types of missing data in detail.

Missing data at the time of diagnosis

A simple way to do diagnosis is to do it only when all of the variables are known. Commonly used classifiers (*e.g.*, neural networks) can then be used. This has been done by many authors, see Section 6.3.1 on page 101. There are however some quite severe drawbacks with this. For example, there may be high costs in finding all of the attribute values, either in time, money or suffering for a patient.

If, for example, our system called for a tissue sample from some internal organ we would have to get that regardless of whether it would be necessary for the diagnosis or not. It is quite obvious that we only wish to find out about the attributes that are absolutely necessary.

Any feed-forward neural network calculates conditional probabilities if the coding is suitable and the error-function and output transfer functions are of certain forms (see Section 6.3.1 (page 101)). They therefore seem suitable for our task. However, they can't take care of missing data since there is no access to the joint probability distribution [Ahmad and Tresp, 1993].

To correctly handle missing data in an ordinary feed-forward neural network model it has been proposed that several different networks each of which have the observed variables as their only inputs are to be used [Sharpe and Solly, 1995; Ghahramani and Jordan, 1994a]. Although this would give satisfactory results when the number of missing variables is low ([Sharpe and Solly, 1995] have only two missing variables), it will not work for a larger number. The reason is that every combination of missing inputs need their own network.

We can not expect any particular order of the variables so questions can be answered in any desired order. This leads to $n!$ networks for n inputs, *i.e.*, an exponential progression. For 5 variables 120 networks are needed, for 10 about 3.6 million! This scheme will therefore not work even for a small number of missing inputs. This is of course a worst case scenario. To reduce this complexity—perhaps only some of the combinations are really missing—very specific knowledge would however be needed.

A better way is to request additional data only when it is needed. By asking questions to the user only the evidence necessary to reach to a conclusion need to be given—provided that the right questions are asked. The system should choose questions based on its current knowledge and can use a cost associated with an observation when doing this selection. This means that most variables will be unknown since we need an answer immediately after each test so that we can select the next one to do, if necessary.

Classification with unknown inputs has been addressed in the neural network context [Ahmad and Tresp, 1993; Tresp *et al.*, 1994; Ghahramani and Jordan, 1994b] based on work in statistics [Little and Rubin, 1987; Little, 1992]. The authors have used mixture models of normal distributions so that closed form solutions to optimal regression with missing data can be formulated.

Missing data in the case database

It is very common that entries in databases are not complete. Variables may not have been registered or have not been available at the time of registration. Merged databases may not have exactly the same variables, so that they only partially overlap. Whatever the reason for this the system must be able to handle it.

There are different types of missing training data. [Tresp *et al.*, 1995; Ghahramani and Jordan, 1994a] discuss this in some detail and have found three types:

Missing completely at random. The probability that a variable is missing is independent of the other variables for the data point.

Missing at random. The probability that a variable is missing may depend on the observed components.

Censored data. Not missing at random at all.

The third case refers to the case when a certain range of other variables make it impossible to get the values of some variables, *e.g.*, if temperature is outside of the range of the thermometer or when some people refuse to state their income in a survey.

In general there is nothing that can be done about data censoring without an inclusion of a model of the actual censoring mechanism into the model. Since this would become very domain specific and would obscure the other modules of the presented diagnosis systems, I have not used any techniques to correct censored data.

With the use of probability theory, data that are missing by the first two mechanisms can be corrected. The joint probability distribution (Section 2.1.5 on page 27) can be used to fill in conditionally expected values. This is one of the main reasons why I have used mixture models for the systems described in Chapter 4 (page 60) and Chapter 5 on page 80 below. The standard alternative for missing data is just to fill-in with the expected value (*i.e.*, the mean of the variable), which is called mean imputation in statistics.

The Expectation-Maximization algorithm which is used for parameter estimation for the mixture models, is interesting in this context since it can be formulated to handle missing data in the training examples [Dempster *et al.*, 1977; McLachlan and Basford, 1988; Ghahramani and Jordan, 1994b; Tresp *et al.*, 1994]. The algorithm was in fact designed with this in mind [Dempster *et al.*, 1977]. This is discussed in Section 4.2 (page 62).

2.1.7 Bayesians and Frequentists

There are two different points of view when uncertainties are discussed in statistics. It is an on-going discussion that deserves to be mentioned.

In the *frequentist* or *objectivist* view probabilities are only considered to be relative frequencies, other interpretations are not possible. This limits the discussion to events that can and have taken place a sufficient number of times, so that a frequency can be computed. The variability in the estimation would otherwise be too high. Probabilities of rare or imagined events, *e.g.*, other life in the universe, cannot be considered.

Bayesians or *subjectivists* on the other hand have a more admissive view, and can also interpret uncertainties as a lack of knowledge about something, *i.e.*, as a degree of belief. They express uncertainties in terms of subjective probabilities. Bayes' rule ((2.2) on page 27) is used to update old assessments in view of new data. Subjective *a priori* probabilities are combined with data to obtain *a posteriori* probabilities. In a future step these *a posteriori* probabilities become *a priori* probabilities, and so on.

The world view of the Bayesians is much wider since it includes everything that the frequentists can consider and more. The debate between these two groups has been going on for a long time. Both of them can consequently be argued for. The problem with the Bayesian theory is what *a priori* probabilities to start with, before any data have been received.

The *a priori* probabilities must be chosen somehow, and it is hard to motivate the choice, since different answers can be the result of different priors, if the number of data points is low. Imagine, for example, a court case where an investigative method is used and where such choices have to be made. With one or a few observations, the result will depend on the priors.

For the purposes of this thesis, I assume that we can view probabilities as degrees of belief, and thus take on the Bayesian point of view, leaving this philosophical discussion aside.

2.2 Modeling Joint Probabilities

As discussed above (Section 2.1.5 (page 27)), the large number of probabilities involved can make exact calculation of conditional probabilities intractable. With n binary variables there are $2^n - 1$ combinations of the variables plus the empty set, *i.e.*, no observation at all. It should be evident that a complete estimation in general is intractable.

However, many of these could be zero or insignificant and this will simplify the estimation problem. Knowing this, we still don't know *which* combinations we can discard without invalidating or degrading the result.

Approximations are thus necessary. There are several ways to do this and two of these are discussed in the following sections. The first one is very simple but not good, the second is much better and allows for general use.

2.2.1 Simple Bayes

By assuming that everything is independent of essentially everything else we get a simple form for the approximation of the complete probability distribution. More precisely, we assume independence between all observations, and conditional independence between the observations given the disorders. There is then a simple equation for the conditional probabilities. This is discussed in detail in Section 3.2.1 on page 43.

These simplifying assumptions are of course not valid since there clearly are dependencies between the symptoms and diseases. Because of this fact, such systems are called *Simple, Naive* or *Idiot's Bayes* in statistics. Although they work surprisingly well in many cases, this is certainly not guaranteed and fully dependent on the data set at hand.

An equivalent one-layer linear neural network can be formulated for a Simple Bayes system [Stensmo, 1991]. Since one-layer networks have limited capabilities (Section 1.4.2 (page 15)), this makes evident the weaknesses of this approximation.

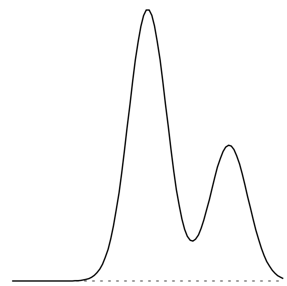

Figure 2.2: Two Gaussians model this curve well.

Probability estimation in this simplified context has been investigated quite extensively and for a long time due to its simplicity and low requirements on computational resources. Some early work is presented in [Ledley and Lusted, 1959; Duda and Hart, 1973] Later work can be found in [Kononenko, 1989; Lansner and Ekeberg, 1989]. I have used it in [Stensmo, 1991; Stensmo *et al.*, 1991; Stensmo *et al.*, 1989]. This work is presented in this thesis in Chapter 3 on page 42.

2.2.2 Mixture models

Finite linear mixtures of probability distributions can be used to efficiently model a joint probability distribution. It is a model that has successfully been used in statistics for some time [McLachlan and Basford, 1988; Dempster *et al.*, 1977].

It is easy to see why it can be useful. The curve in Figure 2.1 (page 26) is readily modeled by a single Gaussian bump. However, in Figure 2.2 that would be impossible without a very complex and unusual description of the function. A sum of two Gaussians on the other hand model it perfectly. This in only a simple one-dimensional example, but you can imagine how it extends to higher dimensions.

It may look very simple to fit two Gaussians to the data in Figure 2.2. Now, imagine the same with noise added, Figure 2.3. It is not immediately

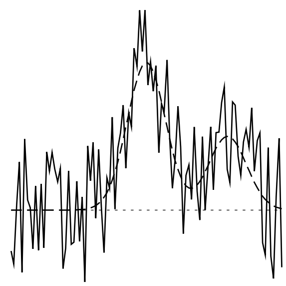

Figure 2.3: Two Gaussians with noise added.

clear where the two functions should be positioned or what their widths should be, or even that two such functions would be suitable. We therefore need a robust method for this. Statistics provides maximum-likelihood estimation methods. They are good at finding the most likely models given the data. Note that the figures do not represent probability distributions.

The Expectation-Maximization (EM) algorithm, described in Section 4.2 (page 62), is such a maximum-likelihood estimation technique. It can be used to efficiently find values for the parameters. Any algorithm that manipulates probabilities must make sure that they are within the required range. These requirements are automatically fulfilled by the EM algorithm [Dempster *et al.*, 1977; Xu and Jordan, 1994]. It can also be shown that mixture models are universal approximators, *i.e.*, any function can be modeled by them [Poggio and Girosi, 1990; Park and Sandberg, 1991].

We have to assume that each data point has been generated by one unique mixture component. This is, however, the only requirement. It translates to that each case in the database has one and only one disease.

2.2.3 Radial Basis Function networks

In the Neural Network community Radial Basis Function (RBF) networks are more common than mixture models which come from Statistics. They are in fact the same thing. If Gaussian distributions are used for the mixture components in a mixture model, it is the same as a Gaussian Basis Function network. If the Gaussian functions moreover are radially symmetric, *i.e.*, the covariance matrix has the same values on the diagonal, and off-diagonal entries are zero, then we have an RBF network.

Early work on RBF models [Poggio and Girosi, 1990; Moody and Darken, 1988; Moody and Darken, 1989] were more or less *ad hoc* in the placement of the centers and widths of the Gaussians. A more principled derivation can be found through maximum likelihood estimation and the EM algorithm [Nowlan, 1990; Nowlan, 1991]. It is the same as for mixture models.

Statistical methods are generally using batch techniques, *i.e.*, all of the training data has to be used at once for the method. Neural network techniques have introduced on-line learning methods [Nowlan, 1991; Ormoneit, 1993], where modification of the parameters can be done after each data point, or every N data points.

Mixture models are used for automated diagnosis in Chapter 4 on page 60 and Chapter 5 (page 80).

2.3 Utility Theory

The previous two sections showed that probability theory is the method of choice for modeling uncertainty and handling missing data. Probability theory can analyze and predict but it cannot tell us how to behave. Based on studies of how people behave, or should rationally behave, utility theory and decision theory have been derived. The work has been based on Game Theory and Economics [von Neumann and Morgenstern, 1947].

Utilities are numbers or values assigned to different states of a system, or to its action consequences, so that the states can be compared to each other. The utilities could be in money (or some otherwise suitable "currency") and indicate preferences for different situations. In a medical context perhaps more natural utilities could be the probabilities of severe complications after some treatment.

It is important to note that utilities are subjective and not objective like

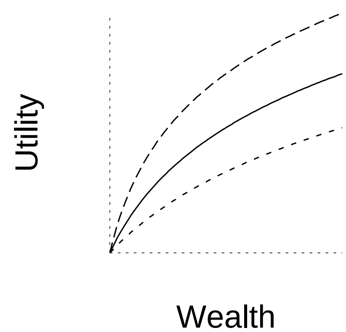

Figure 2.4: Utilities: Some people's preferences for a lottery.

probabilities. Different people will have different preferences for different situations. A billionaire may not be as excited as an average poor graduate student about the prospect of winning a couple of hundred on a lottery. The utility for the student will likely be higher as he runs off to buy some books at the nearest bookstore.

Some patients may want to do risky surgery even though they might die from it, if they can avoid being crippled or some other handicap. Other patients may prefer the opposite. The doctor and the medical insurance company may again have other and perhaps different preferences. A unit called a *micromort* has been used in this context, it is a one in a million chance of death. Another unit is the $QALY$, or quality-adjusted life year, which is equivalent to one year in good health.

By studying different people's preferences in different lottery situations, where the subject is asked to compare simple lotteries with different payoffs, it has been found that utilities are often logarithmic [Grayson, 1960; Russell and Norvig, 1995]. Figure 2.4 shows some examples of this. The gain is much lower at higher wealth (the derivative of $\log x$ is $1/x$). At negative wealth (*i.e.*, debt) the utilities tend to go the other way so as to avoid ruin, Figure 2.5 (over).

Similar experiments have shown that utilities change approximately linearly for small changes of wealth. This means that small lotteries will be quite insignificant and with little risk. We should keep in mind though that

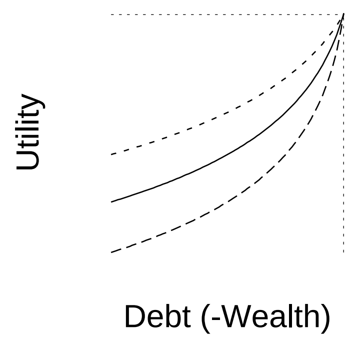

Figure 2.5: Utilities to be expected when in debt.

this is not the case for higher stakes. With the knowledge of this, utilities are usually in practice still modeled as being linear. The reasons for this are probably a combination of ease of implementation, specification and analysis. It is conceptually easy to use linear utilities since they can be specified with the help of an expert, and since there is a clear interpretation of each utility value.

For the diagnosis problem the utilities are the "cost" of misclassification since that could lead to serious consequences. It is important to note that utilities are subjective, in this case it means that a patient and a doctor may have different utilities. The diagnosis system designer must choose an appropriate set of utilities. In the linear case this is a matrix whose entries are the utilities for combinations of misclassifications.

The utilities still have to be specified and this is normally done by hand. Therefore an expert that can assess all needed values correctly is needed. The main theme of this thesis is to avoid manual specification and instead use learning and approximation techniques. One way around this is to use a simple approximation that does not require an expert. This is the approach used in Chapter 4 on page 60.

The results with this approximation are however not always good. A learning technique called Temporal-Difference reinforcement learning was therefore used in Chapter 5 (page 80) to find the correct utility values by working backwards from the quality of a sequence of questions (*i.e.*, a complete diagnosis session) that was decided with the help of the utilities. This technique also allows for the use of non-linear utilities since the learning methods can handle that in the same way as for linear utilities.

See [Russell and Norvig, 1995; Pearl, 1988; Grimmett and Stirzaker, 1992] for a fuller discussion on utilities and their use.

2.4 Decision Theory

In the previous section utility theory was shown to be a way to assign values to different situations so that they can be compared. Decision theory combines probability theory and utility theory so that the utilities can be used for decision making.[2] Expected future utilities are calculated using predicted probabilities for future events whose utilities are known.

Research in Decision analysis started from the theory of games and in Economics half a century ago. [von Neumann and Morgenstern, 1947] formulated its most important idea, the *Maximum Expected Utility Principle*. It states that the most rational behaviour in a situation is the one that will gain the most on average. The gain is measured in expected utility. This quite intuitive principle have also been demonstrated in psychological experiments.

For diagnosis, the decisions we have to make are:

1. What information should we request next out of all unanswered questions and tests?
2. When are we done with the diagnosis? Is it necessary to continue will all of the possible questions?

Decision theory can help us with this.

1. Using utilities for misclassifications we can select the question that will give us the highest expected utility, *i.e.*, the lowest risk for serious misclassification.

[2] [Russell and Norvig, 1995] put it like this: Decision theory = Probability theory + Utility theory.

2. The second question is answered by the fact that we should stop when nothing more can be gained. This happens when the expected utility is zero or negative. It can become negative after the question cost is subtracted.

A specialization of the Maximum Expected Utility principle has to do with selecting what information is most beneficial to acquire in a given situation. It is called the *Value of Information* [Howard, 1966], and is what the utility gain would be with an extra piece of information compared to without it. The idea is to simulate possible future actions, followed by measurement and evaluation of its consequences. It can be shown that this value will always be non-negative. It can never hurt to find out more, but we might not gain anything at all.

After this the cost of the observation is subtracted. The question that will give the highest gain is selected. After the subtraction of the cost the value can be negative, which means that it would not be beneficial to do any additional observations. The diagnosis is then finished.

The use of utility and decision theory can be motivated from many points of views. They have therefore been put forward as general theories for rational human behaviour. There has been a lot of discussion about the validity of this since there is no agreement on whether we actually behave rationally. [Tversky and Kahneman, 1974] claims that we do not, but the psychological experiments that they claim show irrationality have by other authors been said to be invalid for these conclusions [Cohen, 1981].

There is to this date still a lot of controversy about this. It is perhaps better to turn the question around and say that if people don't behave rationally, the more important it is to use decision theory [Howard, 1990; Heckerman, 1995]. [Russell and Norvig, 1995] also discuss the topic of decision theory and how it may or may not apply to human reasoning in more detail.

Decision theory is used for question selection and for the decision when to stop asking questions in Chapter 4 on page 60 and Chapter 5 (page 80).

Chapter 3

Simple Bayes Diagnosis

This chapter describes a system that is based on the simplifying assumption that everything is independent of each other. The result of this is a particularly simple neural network structure and corresponding learning algorithm. It is, however, too simple as discussed in Section 2.2.1 on page 33. Models of this type have therefore been called *Simple, Naive* or *Idiot's Bayes.*

This chapter is included here as a first attempt to build a diagnosis system according to the principles outlined in the previous chapters. The chapter is based on my Licentiate thesis [Stensmo, 1991] and [Stensmo *et al.*, 1991] and is a summary of work I did between 1989-1991.

A system that is much more interesting and that can correctly do diagnosis is presented in the following chapters.

3.1 Simple Bayes

Applying the independence assumptions to a probability model result in a neural network model that is a fully-connected one-layer neural network. It consists of a network layer and a separate input layer, and is thus a one-layer network with its pros and cons, as discussed in Section 1.4.2 (page 15).

Learning the parameters of the network is done with one sweep over the training examples during which statistics are collected. The collected statistics are then used to set the weights of the connections and biases of the units in the network. This one-shot learning is very attractive since it is very fast and independent of any initial values. The result of the learning is that correlated units get positive connections, anti-correlated units get negative connections and uncorrelated units get connections strengths that are zero.

Similar models have been used for a long time starting in the 1950's [Ledley and Lusted, 1959; Kononenko, 1989; Lansner and Ekeberg, 1989]. Other artificial neural models could also have been used for this problem. The chosen model was used because of its interesting properties, most importantly a probability interpretation, that makes it suitable for building a diagnosis system.

3.2 Neural Network Model

3.2.1 The learning rule

A probability distribution over a set of variables can be used to describe the system it models as was discussed in Section 2.1 on page 24. The indirect method for conditional probability calculation has been used for in chapter.

Dependencies can be stated as conditional probabilities between variables. For example, the *a posteriori* probability of an event q in a system described by the variables $\{q, i_1, \ldots, i_n\}$, can be written as the conditional probability

$$p(q|i_1, \ldots, i_n). \tag{3.1}$$

This will be correct if all of the events needed to describe the system are included in the set $\{q, i_1, \ldots, i_n\}$, *i.e.*, the system can be described by $n+1$ variables. Unfortunately, if we want to use (3.1) we have to know the conditional probabilities for all combinations of all values of q and $i_1, \ldots, i_n$. There are 2^{n+1} such combinations if the events are binary. In that case the system would be a kind of a look-up table. This could work for a small N, but is unreasonable otherwise due to the exponential growth.

If we use Bayes' rule[1] to transform (3.1) into the indirect form for conditional probability calculation, it becomes:

$$p(q|i_1, \ldots, i_n) = p(q)\frac{p(i_1, \ldots, i_n|q)}{p(i_1, \ldots, i_n)}. \tag{3.2}$$

This is, however, still not satisfactory since it all the same involves a large number of probabilities.

To make the system computationally tractable we will introduce some independence assumptions that will simplify the calculation and memory requirements to a great extent. We must however keep in mind that these independence assumptions are not correct. They are only valid if our events

[1] The use of Bayes' rule (2.2) on page 27 is the reason why this artificial neural network model has been called a *Bayesian Neural Network* [Stensmo, 1991]. Note that there is another class of models that are more properly called *Bayesian Networks*. They are discussed in Section 6.3.2 (page 102).

really are independent, or at least statistically independent. In reality this is almost never the case unless purely random or artificial data is used. The effects of the independence assumptions are discussed below.

We will now derive a learning rule inspired by artificial neural networks (Section 1.4.2 (page 12)). The following independence assumptions are introduced:

$$\begin{aligned} p(x,y) &= p(x)p(y) \\ p(x,y|z) &= p(x|z)p(y|z) \end{aligned} \tag{3.3}$$

for all units x, y and z. As mentioned above, we can't expect units to be independent in this way, *e.g.*, there are probably always two eyes in a face viewed from the front. Units in a non-random pattern will usually also not be uncorrelated. If some units occur together more often than can be expected from their prior probabilities, *i.e.*, $p(i_1, \ldots, i_n) > p(i_1) \ldots p(i_n)$, then $p(i_1, \ldots, i_n)$ will be underestimated, and the correlation is positive. If the opposite occurs (called negative correlation) the joint probability will be overestimated. Only with equality (*i.e.*, no correlation) is it correct.

Using (3.3) in (3.2) on the page before results in

$$p(q|i_1, \ldots, i_n) = p(q) \prod_{k=1}^{n} \frac{p(i_k|q)}{p(i_k)}. \tag{3.4}$$

As can be seen, this formula contains just a small fraction of the original probabilities. Rewriting (3.4) using logarithms yields

$$\exp(\log p(q) + \sum_{k=1}^{n} \log \frac{p(i_k|q)}{p(i_k)}). \tag{3.5}$$

We will now compare this to a standard formulation of artificial neural networks (Section 1.4.2 (page 12)). The units are assumed to have binary (0/1) output units and input–output functions of the form

$$f(b_q + \sum_{k:i_k=1} w_{q,i_k}) = f(b_q + \sum_{k=1}^{n} i_k w_{q,i_k}), \tag{3.6}$$

for a unit q. f is an output function, normally monotonic and used to limit the output of the unit to within some range. b_q is the *bias* value of unit q

and w_{q,i_k} is the connection *weight* between units q and i_k. This means that we sum only weights from units that have input, *i.e.*, only active units. We can now identify terms in (3.5) and (3.6)

$$\begin{array}{rcl} \log p(q) & \Longleftrightarrow & b_q, \\ \log \frac{p(i_k|q)}{p(i_k)} & \Longleftrightarrow & w_{q,i_k}. \end{array} \tag{3.7}$$

$p(i|j)$ is according to the definition of conditional probability equal to $p(i,j)/p(j)$, so instead of having to remember large amounts of probabilities, only combinations of two and the *a priori* probabilities for the variables are needed. The corresponding weights and bias values can be calculated once these probabilities are known.

3.2.2 Probability estimation

The unconditional and conditional probabilities can be estimated with counters. With N as the number of patterns in the training set the *a priori* probability of an event i can be estimated with a counter c of how many times it occurs out of the total N:

$$p(i) = \frac{c_i}{N}$$

and

$$p(i,j) = \frac{c_{i,j}}{N},$$

so that

$$\frac{p(i|j)}{p(i)} = \frac{p(i,j)}{p(i)p(j)} = \frac{c_{i,j}N}{c_i c_j}$$

for events i and j. Here c_i counts how often a unit i is active, and $c_{i,j}$ counts how often a unit i and a unit j occurs simultaneously.

The bias of a unit i should with this learning rule be set to

$$\text{bias}_i = \log \frac{c_i}{N},$$

as required by (3.7).

There are two singular points that needs to be taken care of: $c_i = 0$ or $N = 0$. The case $N = 0$ can be skipped immediately since learning clearly is not interesting if there is nothing to learn, *i.e.*, the database is empty. When $c_i = 0$ this means that the unit i was never active in any of the patterns to learn. It should thus contribute with 0 in (3.4) (page 44) and $-\infty$ (or some sufficiently large negative number) in (3.5) on page 44.

The weight between a unit i and a unit j is set to

$$\text{weight}_{i,j} = \log \frac{c_{i,j} N}{c_i c_j} \tag{3.8}$$

Singular points are $c_{i,j} = 0, N = 0, c_i = 0$, or $c_j = 0$. $N = 0$ is as for the bias case. If $c_i = 0$ or $c_j = 0$ then the unit i is uncorrelated with unit j and we set the weight to 0. If $c_{i,j} = 0$ but both c_i and $c_j > 0$, then units i and j never occur together. They are anti-correlated and the weights should be $-\infty$. In this manner, correlated units get positive connections, anti-correlated units get negative connections and uncorrelated units get zero connections, as was desired.

Only one look at the training set is needed to set the biases and weights. This is sometimes referred as *one-shot learning*, as opposed to the iterative learning required by other methods, *e.g.*, the back-propagation learning algorithm [Rumelhart *et al.*, 1986a], and other gradient descent variants.

Additionally, if the counters c_i, $c_{i,j}$ and N are kept between learning occasions, we automatically get *incremental learning*, *i.e.*, the learning does not have to start all over again when new additional training data arrives, as is the case with most other methods.

In (3.6) (page 44) it was assumed the the input values were binary. Although our probabilistic interpretation will be lost, a natural generalization to continuous values is to multiply by the output values instead of multiplying by one when a unit is active. The final input–output function, including continuous inputs, for the Simple Bayes artificial neural network model derived from (3.1) on page 43 is

$$o_q = f(b_q + \sum_{k=1}^{n} o_k w_{q,i_k}), \tag{3.9}$$

for a unit q and parameters as discussed above.

Since the independence assumptions (3.3) (page 44) are almost always violated, we cannot expect the output from (3.9) to be in the range $[0, 1]$. To ensure that the requirements for probability variables (Section 2.1.3 (page 26)) are adhered to, we use a limiting function in place of f in (3.9). The output is never allowed to be greater than 1. If the output is less than or equal to the bias value of the unit, the output is set to 0. This is done so that the output will be 0 in a system with no input.

3.3 A Simple Diagnosis System

In this section the neural network model derived in the previous section is used to build a system for automated diagnosis. To be able to do that several issues have to be addressed. Among them are, how to represent variables and how to select which question to ask next.

3.3.1 Representation

There are several possible ways to represent or map observations and disorders onto the units of the network.

One unit per observation and disorder. A simple and natural solution is to use one unit per observation and disorder. This will give sparse patterns (dependent only on observation distributions) and thereby good storage capacity. This is the coding used in Chapter 4 (page 60) and Chapter 5 on page 80. The Simple Bayes system will unfortunately not work with this coding.

Two units per observation and disorder. A slightly more complicated alternative is to use two units per observation and disorder. The additional unit in a pair codes the non-existence of the observation. It is necessary to use this for the Simple Bayes system to function properly. A drawback with this coding is that the patterns will always be fifty percent active which gives the system low storage capacity.

A combination of the two. A combination of these two coding methods could be used, but then there is the choice of when to use which.

In the work reported here, two units are always used to code observations and disorders. We will call the first unit p^+ and the second p^-, for observation p. The unit p^- is called a *negation unit* and is written as the observation name prefixed with a dash ("`-`"), *e.g.*, `bad` is bad$^+$ and `-bad` is bad$^-$, *etc.*

Interpretation of unit outputs

From the probability theory based derivation of the learning rule the unit activities in the model can be interpreted as conditional probabilities (Section 3.2.1 (page 43)), or the *belief* in the output units given the observations.

However, when two units are used to code an observation it is important to note that their activity values are not always the complements of each other. This is because the independence assumptions used in deriving the learning rule are generally violated (see Section 3.2 on page 43).

We must therefore find a way to use the knowledge represented by a unit pair. The negation unit believes that the observation exists with probability $1 - p^-$. The average of this and p^+,

$$\hat{p} = \frac{p^+ + (1 - p^-)}{2}, \tag{3.10}$$

has shown to be a good measure of the belief of an observation. It is somewhat *ad hoc*, but correct if the independence assumptions were not violated since $\hat{p} = p^+$ if $p^- = 1 - p^+$, *i.e.*, the complement is correct. $\hat{p}$ can be called a composite belief value.

3.3.2 Control strategies

Due to the simplicity of the model used in this chapter, the system is sometimes not very robust. Therefore, several different modes of operation had to be designed. Experiments with the described system has shown that three different modes of operation were needed for proper operation.

Finding a hypothesis. The initial questions in a query-reply session are asked in a general way to quickly find a hypothesis. A question generation strategy that quickly discriminates the disorders by choosing questions that splits them into as large groups as possible is used in order to discard a large chunk of them in each step.

Correcting inconsistencies. When inconsistencies occur between the actual user input and the belief of the system they are corrected by asking the user to reconsider his previous reply. Inconsistencies tend to make the query-reply session longer by forcing disorder units to become less activated or inactivated. If there was a hypothesis but it is lost, the initial strategy for choosing questions to find a hypothesis is repeated.

Verification of the current hypothesis. When there is a hypothesis it has been found favourable to try to verify it. Questions are chosen to try to verify the hypothesis by asking for information on specific observations leading to the current goal. A *yes* answer will strengthen the hypothesis and a *no* answer will weaken it. If the hypothesis is lost, the initial strategy for choosing questions to find a hypothesis is repeated.

Finding a hypothesis

Initially in a query-reply session the system usually does not have a disorder hypothesis. It is then a good idea to try to find a hypothesis as quickly as possible by asking a few discriminating questions.

It is important that the query-reply system does not ask too many or irrelevant questions since the necessary investigations may be costly or time-consuming. The questions chosen should be based on current knowledge, *i.e.*, the current belief values of the network units. Several question generation strategies have been tried earlier, see [Stensmo *et al.*, 1989]. It was found that for the system to function well it was best to use knowledge from both units of a unit pair. This way information that may be important is not thrown away.

A good initial strategy is to try to split the input into as large parts as possible, so that the answer will immediately remove many disorders and observations from consideration. One way to do this in our neural network system is to ask about the value of an observation that the system is most undetermined about. In our probabilistic interpretation this means the unit pair whose belief average, $\hat{p}$ defined by (3.10) (page 48), is closest to $\frac{1}{2}$.

The difference between $\hat{p}$ and $\frac{1}{2}$,

$$d(x) = 1 - |x - \frac{1}{2}|, \tag{3.11}$$

has been used as a measure of how determined an observation is, since a probability variable is most undetermined when it is $\frac{1}{2}$, Figure 3.1. Also, solving $\hat{p} = \frac{1}{2}$ gives $p^{+} = p^{-}$, *i.e.*, we should ask about the observation whose positive belief value is close to the belief value of its negation unit. At the time when I did this work, I used this difference as a heuristic. A more principled solution is to use binary entropy.

Shannon's entropies

Entropies [Shannon, 1948] measure the amount of information present, or alternatively the lack of information. For a binary probability variable x the entropy is:

$$H(x) = -x \log_2(x) - (1 - x) \log_2(1 - x). \tag{3.12}$$

The value of this is 0 when x is 0 or 1, *i.e.*, when the value is certain. It is maximized when $x = \frac{1}{2}$. It roughly behaves as measuring the difference to $\frac{1}{2}$, see Figure 3.1, so the heuristic was not bas after all.

If we view the attributes as binary probability variables, with the value one if they are present and zero if not, the variable with the maximum entropy (or the heuristic) is the most undetermined. This observation is then what we need more information about.

There is a drawback with this method. Consider the case when there are completely random variables in the database and that they have $\frac{1}{2}$ as their average values over all instances. They will then always be predicted to be

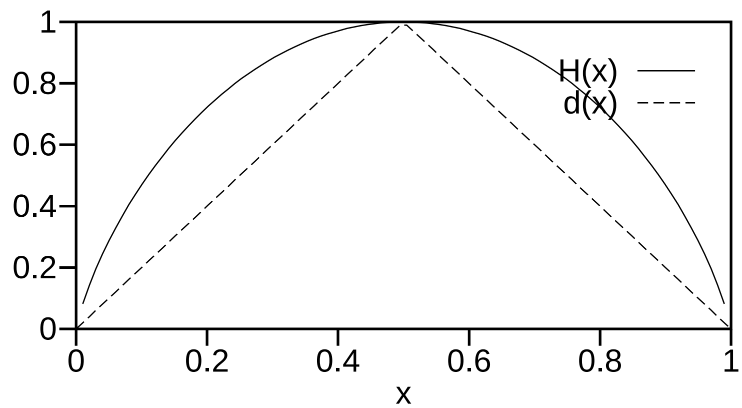

Figure 3.1: Binary entropy compared to heuristic.
The entropy of a binary probability variable x. $H(x)$ is defined in (3.12). In the diagram is also shown the heuristic question valuation function $d(x)$, Equation (3.11), for comparison.

just $\frac{1}{2}$, and thus always be the perfect candidates for the next question. This is quite unfortunate since they are completely irrelevant.

Section 4.5.4 on page 76 shows quantitatively that this result is indeed the case for random data. This is the reason why I did not use this method for the more recent systems presented in Chapter 4 (page 60) and Chapter 5 on page 80. See the former of these chapters for a description of a principled way of question selection based on decision theory.

Correcting inconsistencies

Inconsistencies tend to disturb the query-reply process and make the query-reply session longer. They force disorder units to become less activated or to become inactive. Inconsistencies should be corrected as soon as possible, preferably as soon as they arrive, by asking the user once more about the detected observation. To avoid unnecessary questions, an observation is never re-asked more than once in the current implementation. Typing errors are assumed to have been excluded when the user has already answered a question twice.

Then what is an inconsistency? Several definitions have been tried in simulations. One that has shown to be good in practice is to look at the weight from an input unit to the hypothesis unit. If the magnitude of the weight times the activity is less than zero, then this unit will decrease the activity of the hypothesis unit and is thus a likely candidate for an inconsistency. Choosing the unit with the highest such product has given good results.

Other ways of defining inconsistency include looking at the difference between the belief values of units and comparing them to what input has been given to them (if any). If this is greater than some value (*e.g.*, $\frac{1}{2}$) then there is an inconsistency. However, this did not give as good results as the method above.

If there is a hypothesis and it is lost, the initial strategy for choosing questions to find a hypothesis is restarted.

Hypothesis verification

Once a hypothesis was found it was beneficial to verify it. This is done by asking for information on specific observations of the current goal, by asking the user about the values of observation units that the system believe should be active, *i.e.*, that have high output values, but that may not yet have a belief value of 1.

It was found that asking for observations that have not been answered already, and that have a belief value higher than or equal to the predicted belief of the current hypothesis, was a useful strategy. If it is correct the user will confirm it, otherwise the answer will help the system to reconsider its hypothesis.

If the hypothesis is lost the initial strategy for choosing questions to find a hypothesis is repeated.

3.4 Evaluation

It can be difficult to *evaluate qualitatively* the behaviour of a query based classification system. Usually only an expert knows what is a relevant question in a certain context, and this expert may not always be able to state why a certain question is relevant or not. Different experts may also ask different questions or ask questions in different orders. There is usually also no absolute truth to compare the behaviour of a system to if the problem is complicated.

Quantitative evaluation is a little bit easier to perform, and can be performed in an automated way. Prototype descriptions can be used as a key to evaluate the relevance of a question and the answer to a question. This section presents some results using the described model. Many more results can be found in [Stensmo, 1991].

A number of different test databases were used to test the Simple Bayes system. They are described in Section A.2 (page 114) and reproduced in full in [Stensmo, 1991]. An objection to this choice of databases is that they can not be considered natural. Note, however, that it must be possible to learn the databases with the learning rule described in Section 3.2 on page 43, *i.e.*, the patterns have to be at least statistically independent. The system described in Chapter 4 (page 60) use a real-world database.

When an answer is classified as *right* or *wrong* in this discussion it is not with reference to some objective truth, it is instead in the knowledge domain of the system. The only things the system knows about is what is included in, and can be inferred from, its database. It does not have any other knowledge, for example common knowledge about the world in the manner people do (see also Section 1.3 on page 8). A correct answer does therefore not have anything do with truth, it merely means that an answer is consistent with the current database.

3.4.1 Average number of questions

To measure quantitatively the behaviour we run the system and answer correctly with some probability, one or lower, and incorrectly otherwise. We then keep track of how many questions are needed to reach to the correct conclusion. These numbers are then used to calculate average number of questions needed and standard deviations.

We also keep track of if we get stuck somewhere in an incorrect line of reasoning and cannot get to the correct conclusion at all. The average number of questions will give us a picture of how well the system performs. The variance (or standard deviation) gives an indication of how robust the system is.

To find the minimum number of questions, a decision tree is thought to have been constructed for the problem. The structure of such a tree will depend on the structure of the problem. It is easier to discriminate between dissimilar goals than between very similar ones. Note that the following line of reasoning only applies to trees with binary question nodes. It is assumed that each question moves to a new level, so that questions are unrelated.

There is a most favourable case which gives us the lower limit. Note that it might not be possible to reach the lower limit depending on the structure of the data. This has to do with tree balancing. An unbalanced tree will be very different from a perfectly balanced one.

A theoretical minimum for the average number of questions is achieved if the questions are asked in order to choose branches in a perfectly balanced binary tree. The minimum average depth is then the depth of a balanced binary tree, which is greater than or equal to $\log_2 c$, where c is the number of leaves. It is equal only if the tree is full. This is the best we can achieve if we have perfect knowledge. It is however not certain that it is at all possible to construct this most favourable kind of tree for a data set.

3.4.2 Quantitative evaluation

Table 3.1 shows the average number of questions obtained by using the system described in this chapter when all questions are answered according

Database	Outcomes	Average	St. dev.	C. v.	Min.
Animals	32	5.84	1.09	0.19	5.00
Animals2	32	5.88	1.17	0.20	5.00
Mushrooms	28	5.29	0.75	0.14	4.86
Bumble-bees	48	6.54	1.19	0.18	5.67
Tree	39	6.77	2.11	0.31	5.36
Random2	26	5.23	0.85	0.16	4.77
Random3	23	5.04	1.00	0.20	4.61

Table 3.1: Simulation results for the Simple Bayes model.
Simulation results and theoretical minimum (see Section 3.4.1) for the test databases. The coefficient of variation (C. v.) is the standard deviation divided by the average.

to the prototype description.

As we do not give any erroneous answers in this example, the inconsistency correction mechanism was not used. Verification was also not used. The line of questioning was stopped as soon as the correct answer was reached. This was a test of the quality of the hypothesis finding question generation strategy.

The number of questions needed to reach to a conclusion is remembered and the average and standard deviation is calculated afterwards. The average is compared to the most and the least favourable cases in average number of questions calculated for the different databases in Section A.2 on page 114.

The results show that the number of questions generated by this approach are on the average not far from the minimum. It can be seen that the results do not differ very much no matter what databases are used. The only test database that seem to differ from the other ones is the *Tree* database which has a comparably higher coefficient of variation due to the fact that it is highly unbalanced. Note that it might not be possible to reach to the minimum average number of questions depending on the structure of the problem.

3.4.3 Example of a diagnosis

To demonstrate the qualitative behaviour of this system, to complement the quantitative evaluation, an example is reproduced. For these examples a database with 31 animals described by 81 observations were used. It was unfortunately not the same as in the previous section, but was chosen to be able to compare with a previous study. It is listed in [Stensmo, 1991] where several additional examples also can be found.

In this sample session, Figure 3.2, the system did not receive any initial input data. All belief values of the disorder units were then initially their *a priori* probabilities in the database, in this case all of the animals were equally probably, with a probability of $\frac{1}{31} = 0.0323$. When the query-reply sequence is started the first question is `land-living`. The answer to this will choose between the two main groups of animals in the database.

The output from the system starts with the current hypothesis within square brackets. It will be empty if there is no hypothesis. This is followed by a question, generated by the question generation strategy which depends on whether the system has an hypothesis or not.

Before each question, the current belief values of the disorder units were shown. Belief values less than 0.02 are not shown. The question was then answered by the user, then a new question was generated and so on. A *yes* or *no* were the only answers from the user. The rest of the example was generated by the artificial neural network system.

```
[] LAND-LIVING? yes
0.0625 SKUNK RAT RABBIT PIG LION KANGAROO HYENA HORSE
       GIRAFFE ELEPHANT DOG CAMEL BEAR BAT APE ANTELOPE
[] EATS-GRASS? no
0.0156 RABBIT PIG KANGAROO HORSE GIRAFFE ELEPHANT CAMEL ANTELOPE
0.0883 SKUNK RAT LION HYENA DOG BEAR BAT APE
[] TAIL? yes
0.0245 HYENA BEAR APE
0.0318 CROCODILE
0.0380 PIG KANGAROO HORSE GIRAFFE ELEPHANT CAMEL
0.1766 SKUNK RAT LION DOG BAT
[] BIG? no
0.0226 PIG HORSE GIRAFFE CAMEL
0.0321 HYENA
0.0453 KANGAROO ELEPHANT
0.0464 LION
0.2385 SKUNK RAT DOG BAT
[] MEDIUM? yes
0.0313 LION
0.0536 RAT BAT
0.0702 PELICAN
0.0758 ANTELOPE
0.0970 HYENA
0.1099 KANGAROO
0.6280 SKUNK DOG
[] BROWN? yes
0.0559 PELICAN
0.0617 ANTELOPE
0.0915 RAT
0.0954 KANGAROO
0.1178 SKUNK
0.1330 HYENA
0.6583 DOG
[DOG] DOG?
```

Figure 3.2: Example of a query.

Chapter 4

Mixture Model Diagnosis

In the previous chapter a simple method was used for probability estimation. The previous chapter's question selection method was also not adequate and had to be complemented with other machinery to make it robust. Such methods are thus not adequate for a more complex system.

This chapter presents a general and reliable method for automated diagnosis that is based on probability and decision theory. It is based on [Stensmo and Sejnowski, 1994; Stensmo and Sejnowski, 1995a]. This work was done 1994-1995.

4.1 Mixture Models

As previously discussed in Section 2.2.2 (page 34), a method from parametric statistics called *mixture models* [McLachlan and Basford, 1988] is good for modeling joint probability distributions. The joint distribution is modeled as a combination—or mixture—of component distributions. It has recently been used for supervised and unsupervised learning problems [Nowlan, 1991; Jacobs *et al.*, 1991; Ghahramani and Jordan, 1994b].

A simple closed form solution for an optimal solution to the missing data problem (Section 2.1.6 on page 28) can be achieved with appropriate mixture components. This is one of the reasons why the method is very useful. The method is parametric in the sense that the form of the mixture components have to be predefined. The parameter values are found through an iterative learning algorithm.

A mixture model is best viewed as a kind of *data generative process*. We assume that there was an unknown process that generated all of the data points. The learning task is to find this process. The fact that mixture models are data generative processes will help in the design of the learning method used to find the parameter values. The method that most likely generated the data is sought.

In Section 2.1.5 (page 27) it was discussed that the modeling of joint probability distributions can be intractable unless simplifying assumptions are introduced. In this chapter we assume that each data point was created independently by one of the mixture components. This means that there can only be one disorder or disease per case, or data point, in the database. This assumption is not very severe since most databases have this format. Compared to the assumptions that were used in the previous chapter it is extremely mild.

The mixture components themselves can be of different types, depending on the nature of the data. In this chapter Gaussian and binomial distributions are discussed. They are suitable for continuous and binary variables, respectively. Variables can be transformed so that continuous variables can be interval coded for use with binomial mixture components.

The data underlying the model is a set of N vectors $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, each of dimension D. Each data point $\boldsymbol{x}$ is assumed to have been generated independently from a *mixture density* with M components so that the joint probability is

$$\begin{aligned} p(\boldsymbol{x}) &= \sum_{j=1}^{M} p(\boldsymbol{x}, \omega_j; \boldsymbol{\theta}_j) \\ &= \sum_{j=1}^{M} p(\omega_j) p(\boldsymbol{x}|\omega_j; \boldsymbol{\theta}_j), \end{aligned} \tag{4.1}$$

where each mixture component is denoted by ω_j. $p(\omega_j)$ is the a *priori* probability for mixture ω_j. $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_M)$ are the mixture component parameters. We model the joint probability distribution as discussed in Section 2.1.5 on page 27.

Our goal is to estimate the parameters for the different mixtures so that it is likely that the linear combination of them generated the set of data points. This is accomplished by *maximum likelihood estimation*, a common method in statistics and probability theory. Since the data points are independent, the likelihood is just the product of the probabilities for each of them,

$$L = \prod_{i=1}^{N} p(\boldsymbol{x}_i).$$

Often the logarithm is used since the function results in a monotonic transformation of the data. The product then becomes a sum,

$$\log L = \sum_{i=1}^{N} \log p(\boldsymbol{x}_i).$$

The next section will introduce the learning algorithm that has been used to find the values for the unknown parameters.

4.2 Expectation-Maximization

The *Expectation-Maximization*, or *EM*, algorithm is an iterative maximum likelihood estimation technique that is relatively simple in its final form and converges rapidly. It was first described by [Dempster *et al.*, 1977]. They based their work on more specialized previous work in which they noted that similar methods had been utilized for several different applications.

Two steps are iterated in the algorithm until convergence. The name of the algorithm derives from these steps.

Expectation step. A likelihood is formulated and its expectation is computed in the *Estimation* or *E-step*. For the type of models that we use, this step calculates the probability that a certain mixture component generated the data point in question.

Maximization step. The second step is the *Maximization* or *M-step*. The parameters that maximize the likelihood are found. This can be solved analytically for models that can be written in an exponential form. This is the case for, *e.g.*, Gaussian functions.

The EM algorithm can be sensitive to initial conditions like many other learning methods, though it is much less sensitive then most other methods, *e.g.*, back-propagation [Xu and Jordan, 1994]. Equations can be derived for both batch and on-line learning [Nowlan, 1991].

Initial estimates obtained through the *k-means* algorithm [Duda and Hart, 1973; Tou and Gonzales, 1974] are sufficient to ensure good starting points. The algorithm assigns each data point to the closest of k cluster centers. These centers are then recalculated to be the average values of its data points, and this is repeated until there are no changes, *i.e.*, it has converged. Initially the number of clusters, k, have to be chosen just as the number of mixture components for the mixture models. The centers are initialized to random data points.

Update equations for several distributions, Gaussian and binomial, with and without missing data in the training examples are given below. Details and derivations can also be found in [Dempster *et al.*, 1977; McLachlan and Basford, 1988; Nowlan, 1991; Ghahramani and Jordan, 1994b].

4.2.1 Complete data

From (4.1) (page 61) we form the log likelihood of the data,

$$\begin{aligned} L(\boldsymbol{\theta}|\boldsymbol{X}) &= \log \prod_{i=1}^{N} p(\boldsymbol{x}_i; \theta_j) \\ &= \sum_{i=1}^{N} \log p(\boldsymbol{x}_i; \theta_j) \\ &= \sum_{i=1}^{N} \log \sum_{j=1}^{M} p(\omega_j) p(\boldsymbol{x}_i|\omega_j; \theta_j). \end{aligned}$$

There is unfortunately no analytic solution to the log of a sum in the right hand side of the last equation. However, if we knew which mixture component generated which data point we could compute it.

The EM algorithm solves this via a "trick": A set of binary indicator variables $\boldsymbol{Z} = \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N\}$ is introduced. Each element of $\boldsymbol{Z}$ is a vector $\boldsymbol{z}_i = (z_{i1}, \ldots, z_{iM})$, where $z_{ij} = 1$ if and only if the data point $\boldsymbol{x}_i$ was generated by mixture component j. This is where the assumption that there was only one mixture component that generated each data point is introduced. The variables in $\boldsymbol{Z}$ will allow for an efficient maximization of the likelihood.

As shown in [Dempster *et al.*, 1977; McLachlan and Basford, 1988], the log likelihood can be manipulated into a form that does not contain the log of a sum at all, so that

$$\begin{aligned} L(\boldsymbol{\theta}|\boldsymbol{X},\boldsymbol{Z}) &= \sum_{i=1}^{N}\sum_{j=1}^{M} z_{ij}\log p(\boldsymbol{x}_i,\boldsymbol{z}_i;\boldsymbol{\theta}) \\ &= \sum_{i=1}^{N}\sum_{j=1}^{M} z_{ij}\log\left(p(\boldsymbol{z}_i;\boldsymbol{\theta})p(\boldsymbol{x}_i|\boldsymbol{z}_i;\boldsymbol{\theta})\right). \end{aligned} \tag{4.2}$$

The indicator variable $\boldsymbol{z}_i$ is not known and this means that (4.2) cannot be used directly. Its expectation using the current parameter values $\boldsymbol{\theta}_k$ is used instead. This is the *E-step* of the EM algorithm. It is written

$$Q(\boldsymbol{\theta}|\boldsymbol{\theta}_k) = \mathrm{E}[L(\boldsymbol{\theta}|\boldsymbol{X},\boldsymbol{Z})|\boldsymbol{X},\boldsymbol{\theta}_k]. \tag{4.3}$$

In the E-step, $Q(\boldsymbol{\theta}|\boldsymbol{\theta}_k)$ simplifies to $\mathrm{E}[z_{ij}|\boldsymbol{x}_i,\boldsymbol{\theta}_k]$, see [McLachlan and Basford, 1988; Ghahramani and Jordan, 1994a] for the type of mixture models used.

Now that the expected value is known, the likelihood can be maximized in the *M-step*,

$$\boldsymbol{\theta}_{k+1} = \mathrm{argmax}_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}|\boldsymbol{\theta}_k).$$

This can be done analytically for distributions that can be written in an exponential form. Such model are called *Generalized Linear Models* (GLIM) [Dobson, 1990]). There are many examples of distributions that can be written in this form. Some common ones are: Gaussian, binomial, multinomial and Poisson.

The two steps are iterated until convergence. It can be shown that the likelihood will never decrease after an iteration [Dempster *et al.*, 1977; Xu and Jordan, 1994]. It can stay the same though, indicating a local or global minimum of the likelihood.

4.2.2 Missing data

One of the main motivating ideas behind the EM-algorithm was to be able to handle missing values for variables in a data set in a principled way. Missing data is—especially for diagnosis—a very important problem, as discussed in Section 2.1.6 (page 28).

In the complete data case we introduced indicator variables that helped us solve the problem. They can in fact also be viewed as missing since we do not know their values. Thus, with missing data we just add these missing components to the set $\boldsymbol{Z}$. This composite set is now the set that was missing in the derivation of the EM algorithm [Dempster *et al.*, 1977; Ghahramani and Jordan, 1994b]. We can then proceed as above. To make this more explicit, each vector $\boldsymbol{x}_i$ is split up into two parts, $(\boldsymbol{x}_i^o, \boldsymbol{x}_i^m)$, where o denotes observed data and m missing data. They do not have to be adjacent or ordered, it is just a matter of notation.

The M-step remains as in the complete data case. The E-step becomes

$$Q(\boldsymbol{\theta}|\boldsymbol{\theta}_k) = \mathrm{E}[L(\boldsymbol{\theta}|\boldsymbol{X}^o, \boldsymbol{X}^m, \boldsymbol{Z})|\boldsymbol{X}^o, \boldsymbol{\theta}_k]$$

instead of (4.3).

4.2.3 Mixture components

Normal distributions

Below are given the specialization of the EM algorithm to the case where the mixture components are Gaussian distributions. A Gaussian for mixture component j with mean $\boldsymbol{\mu}_j$ and a full covariance matrix Σ_j is written

$$\begin{aligned} p(\boldsymbol{x}|\omega_j) &= G_j(\boldsymbol{x}) \\ &= \left(\frac{1}{\sqrt{2\pi}}\right)^D |\Sigma_j|^{-\frac{1}{2}} \exp\left[-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_j)^T \Sigma_j^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_j)\right]. \end{aligned}$$

In the general case the covariance matrix is full, *i.e.*, size $D \times D$. Both storage and computational complexity then grows quadratically, $\mathcal{O}(D^2)$, which may de undesirable for large D. The form of the covariance matrix is therefore often constrained to be diagonal, so that the complexity becomes linear, $\mathcal{O}(D)$, or to have the same values on the diagonal, $\Sigma_j = \sigma_j^2 I$. The former one corresponds to axis-parallel oval-shaped Gaussians, and the latter one to radially symmetric Gaussians. A system built with radially symmetric Gaussians is in fact the RBF network that was discussed in Section 2.2.3 (page 35). Such radial and diagonal basis functions have functioned well in many applications [Nowlan, 1991], since several Gaussians together can form complex shapes in the space. With fewer parameters the chance of over-fitting is also reduced, which adds to the benefits of not using a full covariance matrix.

In the radial case the Gaussians have the standard form

$$p(\boldsymbol{x}|\omega_j) = G_j(\boldsymbol{x}) = \left(\frac{1}{\sqrt{2\pi}\sigma_j}\right)^D \exp\left[-\frac{\| \boldsymbol{x} - \boldsymbol{\mu}_j \|^2}{2\sigma_j^2}\right], \tag{4.4}$$

with variance σ_j^2 and mean $\boldsymbol{\mu}_j$ as above.

Complete data. In the E-step the expected value of the likelihood is computed. For the Gaussian case this becomes the probability that Gaussian j generated data point $\boldsymbol{x}$,

$$p_j(\boldsymbol{x}) = \frac{p(\omega_j)G_j(\boldsymbol{x})}{\sum_{k=1}^{M} p(\omega_k)G_k(\boldsymbol{x})}. \tag{4.5}$$

The M-step finds the parameters that maximize the likelihood from the E-step. We will re-estimate the *a priori* probability, the means and the widths of the Gaussians. For time step $t+1$ the update equations become:

$$\hat{p}(\omega_j) \longleftarrow \frac{S_j}{N}, \tag{4.6}$$

$$\hat{\boldsymbol{\mu}}_j \longleftarrow \frac{1}{S_j}\sum_{i=1}^{N} p_j(\boldsymbol{x}_i)\boldsymbol{x}_i, \tag{4.7}$$

$$\hat{\sigma}_j^2 \longleftarrow \frac{1}{DS_j}\sum_{i=1}^{N} p_j(\boldsymbol{x}_i) \| \boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_j \|^2,$$

where

$$S_j = \sum_{i=1}^{N} p_j(\boldsymbol{x}_i).$$

Incomplete data. When input variables are missing the $G_j(\boldsymbol{x})$ in (4.5) is only evaluated over the set of observed dimensions O [Ahmad and Tresp, 1993]. Missing (unobserved) dimensions are similarly denoted U, so that $|D| = |O| + |U|$. (4.4) now becomes

$$G_j^O(\boldsymbol{x}) = \left(\frac{1}{\sqrt{2\pi}\sigma_j}\right)^{|O|} \exp\left[-\frac{\| \boldsymbol{x}^O - \boldsymbol{\mu}_j^O \|^2}{2\sigma_j^2}\right].$$

The update equation for $\hat{p}(\omega_j)$, Equation (4.6) is unchanged. To estimate $\hat{\boldsymbol{\mu}}_j$ we replace the missing dimensions with the prototype dimensions and set $\boldsymbol{x}_i^U = \hat{\boldsymbol{\mu}}_j^U$ and use (4.7). The variance becomes

$$\hat{\sigma}_j^2 \longleftarrow \frac{1}{DS_j} \sum_{i=1}^{N} p_j^O(\boldsymbol{x}_i) \left[\| \boldsymbol{x}_i^O - \hat{\boldsymbol{\mu}}_j^O \|^2 + |U| \hat{\sigma}_j^2 \right],$$

as shown in [Ghahramani and Jordan, 1994a].

Mixtures of binary variables

Assume that each vector $\boldsymbol{x} = (x_1, \ldots, x_D)$ is a D dimensional binary vector. For each mixture component ω_j there is a parameter vector μ_j, so that

$$p(\boldsymbol{x}|\omega_j) = \prod_{d=1}^{D} \mu_{jd}^{x_d} (1 - \mu_{jd})^{(1-x_d)}.$$

This means that the factor μ_{jd} is used if $x_d = 1$ and $1 - \mu_{jd}$ otherwise. This is clearer in the log version:

$$\log p(\boldsymbol{x}|\omega_j) = \sum_{d=1}^{D} [x_d \log \mu_{jd} + (1 - x_d) \log(1 - \mu_{jd})].$$

The E-step is

$$p_j(\boldsymbol{x}_i) = \frac{p(\omega_j) p(\boldsymbol{x}_i|\omega_j)}{\sum_{k=1}^{M} p(\omega_k) p(\boldsymbol{x}_k|\omega_j)}$$

In the M-step we estimate the vector $\boldsymbol{\mu}_j$,

$$\hat{\boldsymbol{\mu}}_j \longleftarrow \frac{\sum_{i=1}^{N} p_j(\boldsymbol{x}_i) \boldsymbol{x}_i}{\sum_{i=1}^{N} p_j(\boldsymbol{x}_i)}.$$

Regression and classification are as in the Gaussian case.

4.2.4 Regression

Least squares regression was used to fill in missing data values during classification. Missing dimensions are denoted U as above, and the completed data is

$$\hat{\boldsymbol{x}}^U = \sum_{j=1}^{M} p_j^O(\boldsymbol{x}^O) \boldsymbol{\mu}_j^U.$$

$p_j^O(\boldsymbol{x}^O)$ is zero if the variable is missing.

The regression is the conditional expectations given the observed variables. In the Gaussian case

$$\mathrm{E}[\boldsymbol{x}^{\mathrm{m}}|\boldsymbol{x}^{\mathrm{o}}] = \sum_{|O|} \boldsymbol{x}^{\mathrm{o}} p(\boldsymbol{x}^{\mathrm{m}}|\boldsymbol{x}^{\mathrm{o}})$$

With missing variables (and radial Gaussians) this is the same as projection onto the known dimensions [Ahmad and Tresp, 1993].

The result of the regression when the disorder variables are missing is a classification, a probability distribution over the disorders. This can be reduced to a classification by picking the disorder with the maximum of the estimated probabilities.

4.3 Decision Theory

During the diagnosis process, the result is refined in each step based on newly acquired knowledge. It it important to select the right questions to ask and to perform the minimal number of tests necessary as discussed in Section 1.2.3 on page 7 and Section 3.3.2 (page 49). In contrast to the simple system used in the previous chapter (Section 3.3.2 on page 49), utility and decision theory is used to select questions, and to decide when to stop asking questions. In Section 7 (page 106) other methods for requesting more information are discussed.

Early work on automated diagnosis [Ledley and Lusted, 1959] acknowledged the problem of asking as few questions as possible and suggested the use of decision analysis for its solution. Decision theory is derived from the theory of games by [von Neumann and Morgenstern, 1947] and is discussed in more detail in Section 2.4 on page 39.

A very important idea from the field of decision theory is the *maximum expected utility principle* [von Neumann and Morgenstern, 1947]. The idea is that a decision maker should always choose the alternative that maximizes some expected utility of the decision. It has been thought that we normally behave according to this principle, or at least should. Our behaviour will then be most rational.

4.3.1 Misclassification utilities

To be able to use decision theory we need to be able to quantify or value each state that the system is in so that different states can be compared to each other. This is done by assignment of utilities to the states. In the diagnosis application we are concerned with not making any errors since the consequences of such can be very expensive, or perhaps life-threatening for a patient.

In the current context we are interested in the *utility of misclassification* since misclassification inevitably will lead to severe consequences for all diagnosis, medical and machine. Each pair of outcomes therefore is assigned a *utility value* $u(x, y)$ that measures the cost of misclassification when the correct diagnosis is x but y has been incorrectly determined. The result is an array of such values. It does not have to be symmetric since there is nothing that says that it is equally bad to mistake a flu for meningitis or the other way around.

These utility values have to be assessed manually in what can be a lengthy and complicated process. For n disorders there are $n(n-1)/2$ combinations, and it thus grows $\mathcal{O}(n^2)$. It can also be hard to come up with appropriate values. For this reason a simplification of this function has been suggested by [Heckerman *et al.*, 1992]: The utility $u(x, y)$ is 1 when both x and y are benign or both are malign, and 0 otherwise. This simplification has been claimed to work well in practice [Heckerman *et al.*, 1992].

With the use of this approximation, the amount of utility assessments has thereby been greatly simplified. They are thus trivially known so that no expert's assessments are necessary. This is appealing from the point of view of this thesis since I strive not to use explicit expert knowledge. Instead of using this simplified model we could have spent a lot of time and effort interviewing an expert to find the appropriate utility values. In Chapter 5 (page 80) a way to elicit the utility values automatically instead of requiring them to be specified manually is presented. It also allows for an extension from linear to non-linear utility models.

There is generally a cost associated with each possible test. The goal is to minimize the total cost and at the same time to quickly come to the correct conclusion. In a medical context we might not of ethical reasons want to use costs if the objective is to make the patient well (although this certainly happens anyway). Instead we want weight in the implications for the patient in terms of inconvenience, pain or risk of future complications.

4.3.2 Myopic approximation

There is another complication with the maximum expected utility principle that can make its exact evaluation intractable. In the ideal case we would evaluate every possible sequence of future choices to see which would be the best. Since the size of the resulting search tree of possibilities grows exponentially this may not be reasonable.

A simplification [Gorry and Barnett, 1967] is to only look ahead one or a few steps at a time. This nearsighted or *myopic* approach has been tested in practice with good results [Gorry and Barnett, 1967; Heckerman *et al.*, 1992].

4.3.3 Inconsistency correction

A similar hypothesis inconsistency correction mechanism to the one that was used in Section 3.3.2 on page 51 has been used in this system.

Before a question is asked by the system it is checked whether the previously given answers are consistent with the current hypothesis. This is done in the following way. Each previously answered question is considered unknown again. It is then filled-in with its expected result given what else is known to find the conditional expectation. This result is then compared to the actual excluded value.

The answer is considered potentially inconsistent if the difference is large, and a normal deviation otherwise. A difference of $\frac{1}{2}$ has been used. When there is an inconsistency, that question is asked again, but only once since conflicting answers could be correct and not erroneous at all. If the user has been asked to check it once, it is considered correct.

With this procedure there has to be a decision threshold over which it is considered an inconsistency and under which it is not. It is possible to get away from this parameter too by looking at the conditional variance in addition to the conditional expectation. Statistical hypothesis testing can then be used to assess whether a value is abnormal or not. See also the discussion on conditional variance in Section 7.3 (page 110) for a possible alternative method.

4.4 A Complete Diagnosis System

In the sections above the building blocks for a complete diagnosis system have been described. It is now time to put them together. The complete system has two phases. First there is a *learning phase* where the probability model is built using Expectation-Maximization learning on a mixture model.

This model is then used for inference in the *diagnosis phase*. Chapter 5 on page 80 describes an additional phase where utility values can also be adapted after each sequence of questions. The system used in this chapter used fixed 0/1 utilities as discussed in the previous section.

4.4.1 Learning phase

The joint probability distribution of the data is modeled using mixture models. Parameters are determined by the EM algorithm. The k-means algorithm is used for initialization of the parameter values before the EM-algorithm is used. These steps make up the learning phase where the model is built.

Variable and disorder variables for each case are combined into one vector per case and form the set of training patterns. Since there are no differences between input and output variables in a mixture model, the model can be used for both finding the diagnosis and for question selection.

4.4.2 Diagnosis phase

When the training of the mixture model parameters has been done the system is ready for use. Sequential diagnosis is then performed in the diagnosis phase. It consists of the following steps and use the model that was built in the learning phase and the utility model that was described above. Figure 4.1 shows a flowchart of this phase.

1. Any initial observations can be entered. Compare this to, *e.g.*, a decision tree where only some initial data can be used, depending on whether it occurs close to the root of the tree or not.

2. Regression is used to find expected values of all currently unknown variables (Section 4.2.4 on page 67). The unknown dimensions are filled in using
$$\hat{\boldsymbol{x}}^U = \sum_{j=1}^{M} p_j^O(\boldsymbol{x}^O)\boldsymbol{\mu}_j^U.$$
This is a rigorous way of handling missing data. A conditional expectation $E[\boldsymbol{x}^U|\boldsymbol{x}^O]$ is found instead of an unconditional one, the *a priori* distribution $E[\boldsymbol{x}^U]$.

3. The Maximum Expected Utility Principle Section 2.4 (page 39) is used to find the expected utility for each possible next question. Subtract the cost of the observation if applicable from the expected utilities. The result are the expected gains.

   Stop if it is determined that nothing would be gained by making an additional observation. This happens when the expected gains all are zero or negative.

4. Recommendation of the most valuable question to answer. The user is asked to determine the correct value for this recommended observation. The user can always choose to do any other observation instead of, or in addition to the recommendation. He is not forced to answer the question as in, e.g., a decision tree.

5. Continue with step 2.

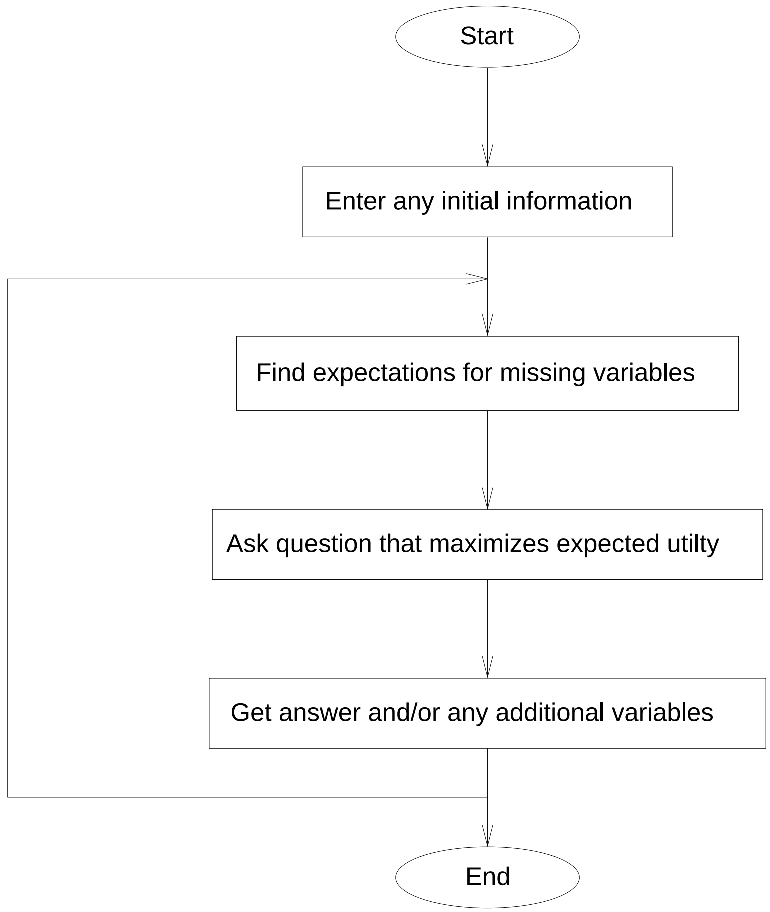

Figure 4.1: Flowchart of the complete automated diagnosis system. This flowchart shows the basic operation of the mixture model system in the diagnosis phase. When the value of a variable is requested (*i.e.*, the answer to a question), any number of questions can in fact be answered. It is not necessary to answer the suggested question. This sequence of questions and answers are repeated until nothing more would be gained by asking an additional question, or when there is a satisfactory separation between the output disorders, or when the user decides to stop anyway.

4.5 Evaluation

This section presents results from numerous experiments performed on the system described in this chapter. These experiments have been done to explore the model's possibilities and to verify that it does what should be expected from theory. The Cleveland heart disease data set which is described in Section A.1 on page 114 was used.

Since the model described in this chapter has superior characteristics compared to the Simple Bayes model, almost any kind of database can be used with it. There is one assumption, however, that is important, and that is that there can only be one disorder for each case in the database. This comes from the way the Expectation-Maximization algorithm was derived.

4.5.1 Model specification

Nominal variables, *e.g.*, the disorders were coded as 1 of N. This means that there was one binary variable for each possible value, with a 1 in the position for the current value to be coded and zeroes otherwise. Continuous variables were interval coded. For the experiments in this chapter, the coding was based on how many standard deviations the value was removed from its mean. In this new coding the variables represent probabilities of occurrence within the different resulting ranges.

This choice of coding was motivated by the approximate normal distribution of the continuous variables in the database. Age is of course not normally distributed since it cannot be negative, but this is a sufficient approximation. In total the 14 variables were coded with 55 units. Gaussian mixture components were used for the experiments in this chapter.

Note that no background knowledge about the domain have been supplied and this diagnosis system is thus, in contrast to, *e.g.*, rule-based expert systems, completely domain independent. Starting values for the means and the widths of the Gaussians were obtained by the k-means algorithm as discussed above.

Thereafter the EM algorithm steps were repeated until convergence.

This took about 60–150 iterations of the two steps, depending on the system configuration. A varying number of mixture components, ranging from 20 to 120, were tried. The best result was then selected for future use in the diagnosis phase.

4.5.2 Quantitative evaluation

Previously reported results for this database have used only presence or absence of the heart diseases. The best of these has been a classification rate of 78.9% with a system that incrementally builds prototypes [Gennari *et al.*, 1989]. 78.6% correct classification was achieved with 60 radial Gaussian mixtures as described in previous sections. This result shows that radial Gaussian mixture models can form good classifiers.

Using all five disorders, instead of only presence and absence, only 66.6% correctly classified patterns could be achieved. Note that previous investigators of this data set have pointed out that there is not enough information in the thirteen variables to completely classify the disorders [Gennari *et al.*, 1989]. I still chose to use this database for benchmark purposes and since there are not many databases of this type freely available.[1] In the next chapter it is demonstrated that these results still can be improved upon by a different choice of coding and mixture components.

4.5.3 Example session

To show how the complete system works in practice an annotated transcript is shown in Figure 4.2 on page 77. In this example, all of the variables are initially unknown, which is a reasonable starting point. Their unconditional prior probabilities are then used as starting values for the diagnosis.

In this example it is shown how the system becomes more and more certain of its conclusion. This can be measured by looking at the *entropy* of the disorder units. Entropy is defined

$$H = -\sum_i p_i \log p_i. \tag{4.8}$$

[1]See discussion in Section 7.2.3 (page 108).

When only one of the disorder units have the value 1 and the other ones have the value 0, the entropy will be zero. Otherwise it will be positive. It may not be possible to reach to the state where the entropy is zero, *e.g.*, if there was noise in the case database.

4.5.4 Quantitative evaluation on artificial databases

In the previous chapter (Chapter 3 (page 42)) a number of smaller test databases were used to evaluate that system. I have also tested them in this system for comparison.

These databases can trivially be learned by mixture models. By using one mixture component per data point we achieve full accuracy, *i.e.*, an error rate of 0%. This result is included to point out that the databases in the previous chapter are not very hard to learn. Still, the Simple Bayes model had problems with some of them.

The average number of questions needed for diagnosis and classification using the techniques used in the previous chapter is shown in Table 4.1 on page 78. Compare this table to Table 3.1 (page 55). The average number of questions are clearly lower for the natural databases. The variances are also much lower.

The second of the random databases, however, actually has a larger average. This is due to the fact that completely random events confuse the entropy based question selection mechanism, as discussed in Section 3.3.2 on page 49. This is quite an interesting result since it seems that the Simple Bayes model in some way is able to compensate for that.

The leftmost number of the five numbers in a line is the estimated probability for no heart disease, followed by the probabilities for the four types of heart disease. The entropy of the diagnoses are given at the same time as a measure of how decisive the current conclusion is. A completely determined diagnosis has entropy 0. Initially all of the variables are unknown and starting diagnoses are the unconditional prior probabilities.

```
Disorders (entropy = 1.85):
  0.541254  0.181518  0.118812  0.115512  0.042904
What is cp ? 3
```

The first question is *chest pain*, and the answer changes the estimated probabilities. This variable is continuous. The answer is to be interpreted how far from the mean the observation is in standard deviations. As the decision becomes more conclusive, the entropy decreases.

```
Disorders (entropy = 0.69):
  0.888209  0.060963  0.017322  0.021657  0.011848
What is age ? 0

Disorders (entropy = 0.57):
  0.91307619  0.00081289  0.02495360  0.03832095  0.02283637
What is oldpeak ? -2

Disorders (entropy = 0.38):
  0.94438718  0.00089016  0.02539957  0.02691099  0.00241210
What is chol ?  -1

Disorders (entropy = 0.11):
  0.98848758  0.00028553  0.00321580  0.00507073  0.00294036
```

We have now determined that the probability of no heart disease in this case is 98.8%. The remaining 0.2% is spread out over the other possibilities.

Figure 4.2: Mixture model system example.

Database	**Outcomes**	**Average**	**St. dev.**	**C. v.**	**Min.**
Animals	32	5.03	0.31	0.06	5.00
Animals2	32	5.06	0.44	0.09	5.00
Mushrooms	28	5.73	0.64	0.11	4.86
Bumble-bees	48	4.86	0.36	0.07	5.67
Tree	39	4.81	0.57	0.12	5.36
Random2	26	4.87	1.01	0.21	4.77
Random3	23	6.56	1.98	0.30	4.61

Table 4.1: Simulation results for mixture models.
Simulation results and theoretical min for the test databases presented in Section A.2 (page 114). The coefficient of variation (C. v.) is the standard deviation divided by the average.

Chapter 5

Reinforcing Utilities

In this chapter the mixture model diagnosis system in the previous chapter is extended with adaptation and learning of the utility values by the use of the Temporal-Difference reinforcement learning algorithm. The utility values are, as in the previous chapter, used for question selection and to decide when to stop asking questions. This chapter is based on [Stensmo and Sejnowski, 1995b]. The work was done in 1995.

5.1 Introduction

Probability theory represents and manipulates uncertainties in a principled way, but it cannot tell us how to behave or what decisions to make. For that we need *utility theory*, which values the usefulness of different states so that they can be compared. Together these two form *decision theory* which deals with how to make rational decisions under uncertainty. See Section 2.4 on page 39 for a longer discussion.

The most important idea in decision theory is the principle of *maximum expected utility* [von Neumann and Morgenstern, 1947]. It is a general principle for rational behavior, and we will in the long run benefit the most if we make our choices following this principle with some utility function. Even "irrational" behavior can in fact be seen as rational—it is the utilities that are wrong. See Section 2.3 (page 36) and Section 2.4 on page 39 for a general discussion on these topics.

There have been a lot of work on how to learn probability models—be it by mixture models or artificial neural networks, or otherwise—but not much on how to learn utility models. Just as with rule-based expert systems or probabilistic networks, they are assumed have been manually crafted by some knowledge engineer.

The utility models are generally assumed to be linear to make their specification easier. Each value will then have a direct interpretation. This chapter shows that learning and adaptation of utility values can be done with reinforcement learning. Even though linear models are used in this chapter, the methods used can readily be extended to non-linear utility models.

A complete automated diagnosis system can be built where the probability model is found from a database of cases, while the utility model can be elicited by asking an expert using the system how he perceives the quality of the line of questions and the quality of the final result. The answer from the expert is the reinforcement signal that will be used by the learning algorithm.

An alternative way of using such a system is to have the system compare the resulting diagnosis of a randomly selected diagnosis sequence, where all of the questions are answered according to the case at hand, with the correct value. The learning will then change the utility values so that the result is correct and this will result in a more robust system, perhaps at the expense of a longer diagnosis sequence. This is demonstrated in this chapter.

5.1.1 Adapting the utilities

As in the previous chapter, decision theory was used to find the most informative question to ask to get more information and to decide when the diagnosis is finished (Section 4.3 (page 68)). Utilities determine the goodness of a misclassification, and the goal is to maximize the expected utility.

In the previous chapter these utilities had to be manually specified. A simple approximation was used. This same approximation has been reported to work well in other work ([Heckerman *et al.*, 1992]), but we found that the results on our database were not optimal in renewed experiments. This made us look for a way to learn the utilities instead of using the approximate values.

Utilities cannot be found directly since they are intricately involved in every question selection in the sequence of questions that make up a diagnosis. After each diagnosis, however, we can get a "grade" from the user of the system, or the system itself, on the quality of the sequence of questions. This grade or reinforcement can be used to elicit better utility values with the help of reinforcement learning.

Reinforcement learning is designed for this situation. In a temporal-credit assignment problem, it is not known where in the sequence of decisions the wrong one was made, for example, in a chess game where a sub-optimal move was made somewhere among all of the moves. Not until the game is over will we know the final result of a move and have to figure out what, out of all of the moves that were made, that were the bad ones, and how they should be corrected.

Reinforcement learning methods makes it possible to use a final reinforcement signal and to distribute it over all of the decisions in the sequence. It has been applied to numerous problems, from the automatic balancing of an upright pole [Barto *et al.*, 1983] to playing winning backgammon [Tesauro, 1992; Tesauro, 1993] to the scheduling of an elevator [Crites and Barto, 1996], and much more.

Several reinforcement learning methods have been developed. One of the best known is the method of *Temporal-Differences* [Sutton, 1988]. It is the method that is used in this chapter. It is explained below.

This chapter demonstrates that utilities can be improved through reinforcement learning. This also means that utility values can be elicited from an expert that gives feedback on the quality of the diagnosis that the system just performed. The expert therefore does not have to know what the utility values are. The only requirement is that he is consistent in the kind of feedback that he gives to the system.

There is reason to believe that this method can also be used to find utility models for other decision theoretic systems, unrelated to diagnosis. The presented method makes it possible to use non-linear utilities in addition to linear ones. This is further discussed in Section 7.3 on page 109.

5.2 The Model

5.2.1 Probability density model

Similar to the previous chapter's model, the joint probability density of the data was modeled by a finite linear mixture model [McLachlan and Basford, 1988] as summarized below. The joint probability density is then

$$p(\boldsymbol{x}) = \sum_{j=1}^{M} p(\boldsymbol{x}, \omega_j; \boldsymbol{\theta}_j) = \sum_{j=1}^{M} p(\omega_j) p(\boldsymbol{x}|\omega_j; \boldsymbol{\theta}_j).$$

There are M mixture components each denoted by ω_j. For each component there are two parameters, the *a priori* probability, or mixing proportion $p(\omega_j)$, and the mixture component specific parameters $\boldsymbol{\theta}_j$.

We have in this chapter used binomial mixture components instead of the Gaussian components that were used in Chapter 4 (page 60). Other components, e.g., multinomial or Gaussian could also have been used (see Section 4.2.3 on page 65).

Each data point $\boldsymbol{x}_i$ out of a set of size N is a D-dimensional binary vector. There is a parameter vector μ_j, also of length D, for each mixture component. Binomial mixture components have the form

$$p(\boldsymbol{x}|\omega_j) = \prod_{d=1}^{D} \mu_{jd}^{x_d} (1 - \mu_{jd})^{(1-x_d)}. \tag{5.1}$$

The Expectation-Maximization (EM) algorithm, Section 4.2 (page 62), was used to find the maximum likelihood estimates of the parameters. The algorithm consists of two steps that are repeated: An Expectation (E) step that estimates the probability that mixture component ω_j generated the data point,

$$p_j(\boldsymbol{x}_i) = \frac{p(\omega_j) p(\boldsymbol{x}_i|\omega_j)}{\sum_{k=1}^{M} p(\omega_k) p(\boldsymbol{x}_k|\omega_j)},$$

and a Maximization (M) step that updates the parameters

$$\hat{\boldsymbol{\mu}}_j \longleftarrow \frac{\sum_{i=1}^{N} p_j(\boldsymbol{x}_i) \boldsymbol{x}_i}{\sum_{i=1}^{N} p_j(\boldsymbol{x}_i)},$$

$$\hat{p}(\omega_j) \longleftarrow \frac{\sum_{i=1}^{N} p_j(\boldsymbol{x}_i)}{N}.$$

Regression was used to find the expected values of the unknown variables. It is the conditional expectations of the missing variables given the observed ones. See Section 4.2.4 on page 67 for more information. In the binomial case this means that $p_j(\boldsymbol{x})$ is only evaluated at the observed dimensions, and that corresponding $\boldsymbol{\mu}_j$ values are used where there are missing dimensions for $\boldsymbol{x}$ in (5.1) on the page before [Ghahramani and Jordan, 1994b].

5.2.2 Decision model

In the diagnosis application we are concerned with misclassification. This is reflected in the utility model. The utilities used for this application are the implications for misclassifications. A linear utility model was used as is customary in decision theory.

The misclassification utilities are in a matrix U, where each entry U_{ji} is the utility of having misclassified outcome i for j. It does not have to be symmetric. In a linear model, the expected utilities of the different outcomes are $u = Up$. The utility for one particular outcome i is then $u_i = \sum_j u_{ji} p_j$. The predicted probabilities for the different outcomes are in p.

The selection of the most informative question to ask was done according to the *value of information criteria* (see Section 2.4 on page 40). First, the maximum expected utility value is noted before any additional observations. Then each possible new answer to the unknown variables are imagined to be tested. For each of them the probabilities of the outcomes and their expected utilities are calculated.

The gain in utility is the maximum expected utility minus the new utility of the best outcome without the new observation. This is calculated for each new variable and is weighed by the predicted probability of occurrence for each answer. This is then the value of information if we observe the variable.

The cost of the hypothetical observation is then subtracted resulting in the *net value of information*. There are established procedures to convert costs (in, *e.g.*, dollars) to units of probability (of, *e.g.*, losing your life as the result of a treatment) [Howard, 1980; Heckerman *et al.*, 1992].

If any of the variables now has a positive net value of information, the one with the highest value is requested. Should this not be the case, nothing more can be gained and we are finished with our diagnosis. If several possible questions have the same value, one is picked randomly. This is perfectly fine since they have the same "values" to the user.

Just as in the previous chapter we simplify the look-ahead by only studying one forward step, the myopic approximation, see Section 4.3 (page 68). Even though it is an approximation it has been reported to work well in many instances [Gorry and Barnett, 1967; Heckerman *et al.*, 1992].

The main drawback with this theory is that we have to get the utility values from somewhere. They have to be assessed by an expert on the domain, which can require a considerable effort. Approximations are therefore often used here too, for example, [Heckerman *et al.*, 1992] used utility values 1 for the misdiagnosis between either two malign or two benign diseases, and the value 0 otherwise.

However, our experiments have shown that this is not optimal as will be seen in Section 5.3 on page 87. The next section presents a way around this problem.

5.2.3 Reinforcement learning

The Temporal-Difference (TD(λ)) algorithm [Sutton, 1988] was used for reinforcement learning of the utility values. Input data are sequences of values for each time step $x_1, \ldots, x_m$. The actual reinforcement z is defined to be data point P_{m+1}, see Figure 5.1 (over). For each time step t a prediction P_t of the final reinforcement z is estimated.

P_t is the prediction function, a function of inputs x and weights w. In the linear case it is just $P_t = wx$. This corresponds to the expected utilities Up. In this way we see how we can rewrite our reinforcement learning equations to a form that can be applied to the utilities.

The temporal-differences algorithm TD(λ) updates the weights by

$$\Delta w_t = \alpha(P_{t+1} - P_t) \sum_{k=1}^{t} \lambda^{t-k} \nabla_w P_k, \tag{5.2}$$

where α is the learning rate, and λ is how past experience is weighed in. [Sutton, 1988] derives and explains this in detail and proves convergence for $\lambda = 0$. [Dayan, 1992] proves convergence for general λ. The method is also discussed in [Haykin, 1994].

$$\begin{array}{rcccc} \text{Data points:} & x_1 & \ldots & x_m & \\ & & & \Downarrow & \\ \text{Expected future rewards:} & P_1 & \ldots & P_m & P_{m+1} = z \end{array}$$

Figure 5.1: Data for Temporal-Difference reinforcement learning. Input data are the sequence of points $x_1 \ldots x_m$. The respective predictions are $P_1 \ldots P_m$. The actual reinforcement is z, which is defined to be P_{m+1}. The arrow denotes the function $P(x, w)$, where w are parameters that define the function. The goal of the learning is to change this function so that the predictions become perfect at each step.

In an on-line intra-sequence version of the TD algorithm [Sutton, 1988] that is appropriate for our application, the sum in (5.2) on the preceding page is performed separately in the eligibility trace variable e_t. This variable makes it possible to do the summation within the sequence. We don't have to wait until afterwards to do the adjustments.

If

$$e_t = \sum_{k=1}^{t} \lambda^{t-k} \nabla_w P_k,$$

then it can be shown [Sutton, 1988] that (5.2) on the page before can be rewritten to two equations

$$\begin{aligned} e_{t+1} &= \nabla_w P_{t+1} + \lambda e_t \\ \Delta w_t &= \alpha(P_{t+1} - P_t) e_t \end{aligned} \tag{5.3}$$

The following update equations were used. Note the change of variables from the standard one to one that conforms with the presentation of our problem. w_t is replaced by U, x_t by p and x_{t-1} by p'.

After each question (*i.e.*, time step), the utility matrix U was updated by

$$\Delta U = \alpha U (p - p') e,$$

and e was set to

$$e = p + \lambda e. \tag{5.4}$$

At the end of the diagnosis sequence when the reinforcement z is received from the expert that evaluates the sequences, U is then updated by

$$\Delta U = \alpha U(z - p)e.$$

and e is set to (5.4) as before.

The reinforcement given in our system was the correct diagnosis for each case, with a one for the correct position and a zero otherwise. Other reinforcement values can also be used as discussed above.

The patterns were presented randomly to remove possible ordering bias in the database. A complete randomized presentation of the entire data set was called one iteration.

We reset the diagonal values to one after each of the iterations for normalization. The identity misclassification, which of course is not a misclassification at all, is then by definition always one. Experiments without this operation still worked but resulted in a larger average number of questions, as well as a higher error rate. It also obscured the resulting utility matrix in the experiments.

5.2.4 The system

In Section 4.4 (page 71), a complete diagnosis system was presented. It had a *Learning phase* and *Diagnosis phase*. The reinforcement learning of the utilities is performed during the Diagnosis phase. The feedback is given after each sequence, but the adjustments are performed after each question presentation.

5.3 Results

The Cleveland heart-disease database, described in Section A.1 on page 114, was used to test the method. It consists of 303 cases where the disorder is one of four types of heart-disease or its absence. There are thirteen variables as shown in Table A.1 (page 115). Continuous variables were converted into a

1-of-N binary code. Nominal and categorical variables were coded with one unit per value. In total 55 binary variables coded the 14 original variables.

To find the parameter values, the EM algorithm was run until convergence. A mixture model with 98 binomial mixture components gave the best classification results using just the mixture model to predict the probabilities of the outcomes from the observations of each case. We tried from 40 to 120 mixture components. The classification error was 46 or 15.2% with 98 components.

This result is much better than the 101 errors (33.3%) reported in the previous chapter and [Stensmo and Sejnowski, 1994; Stensmo and Sejnowski, 1995a]. This is due primarily to a better coding of the continuous variables, but also to the use of binomial mixture components instead of Gaussian ones.

I did not seek to maximize the correctness performance in the previous chapter since the result was already on par with previously reported results. The main aim there was to demonstrate the method and not to maximize performance on that particular data set. It is pleasing to see that the previous result could be improved upon though.

Next the model was tested to see if it works for diagnosis sequences. Using the full utility/decision model and the 0/1-approximation for the utility matrix (left part of Figure 5.2 on page 90), there were 78 errors. Over the whole data set an average of 5.31 questions were used with a standard deviation of 1.77. Note that this is without TD-learning, *i.e.*, $\alpha = 0$. This is thus the same method that was used in the previous chapter. The large number of errors show that the approximation is far from optimal.

To explore the system with TD-learning, λ was varied from 0 to 1 in increments of 0.1. It was found that $\lambda = 0.1$ gave the best result. The learning rate α was also varied over several orders of magnitude. Results for $\alpha = 0.01$ are shown in Table 5.1.

It was found that $\alpha = 0.01$ gave the best result for $\lambda = 0.1$. A higher learning rate gave rise to large fluctuations and instability. A lower learning rate resulted in no improvement of the result but took longer time run.

λ	**Iteration**	**Average**	**St. dev.**	**Errors**
0	6	6.44	2.19	46
0.1	8	5.95	2.39	46
0.2	5	6.44	2.22	46
0.3	5	6.41	2.25	46
0.4	4	6.41	2.20	47
0.5	4	6.32	2.24	47
0.6	1	6.27	2.25	47
0.7	1	6.30	2.22	46
0.8	1	6.26	2.15	47
0.9	4	6.42	2.20	47
1	did not converge			

Table 5.1: Results of TD-learning of the utilities, λ varied. Parameter values: $\lambda = 0.1$ to 1.0, step 0.1. $\alpha = 0.01$.

Simulation results are shown in Table 5.2 on the following page for the best parameter values. The initial utility matrix is shown in the left side of Figure 5.2 (over). The utility matrix after 8 iterations through the entire data set is shown on the right. Note that these results were measured on the past 303 presentations during which learning was done.

As can be seen in Table 5.2 on the following page, the price paid for increased robustness is an increase in the average number of questions. This is perhaps to be expected and show that the system does work. Note that we can in fact get the same accuracy as we had when we used all of the input variables by using only less than half of them on average. At the same time we make sure that we use all of the resources in an efficient way.

It is also worth noting that the final utility matrix has values that are not what we would have guessed, and quite away from the initial values. This serves to illustrate that the problem is non-trivial.

Iteration	**Average**	**St. dev.**	**Errors**
initial	5.31	1.77	78
1	6.32	2.23	47
2	6.25	2.25	45
3	6.37	1.92	51
4	6.49	2.13	47
5	6.48	2.17	47
6	6.39	2.25	46
7	6.16	2.31	49
8	5.95	2.39	46

Table 5.2: Results of TD-learning of the utilities, best results. Parameter values: $\lambda = 0.1$ and $\alpha = 0.01$.

$$\begin{array}{ccccc} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \end{array} \qquad \begin{array}{ccccc} 1 & 0.3322 & 0.3419 & 0.3369 & 0.3090 \\ 0.1158 & 1 & 0.0988 & 0.1387 & 0.2872 \\ 0.0826 & 0.0819 & 1 & 0.0833 & 0.2396 \\ 0.0911 & 0.0649 & 0.0880 & 1 & 0.2479 \\ 0.0360 & 0.0281 & 0.0300 & 0.0826 & 1 \end{array}$$

Figure 5.2: Initial and final misclassification utility matrices. Left: Initial utility matrix. Right: After 8 iterations with parameter values $\lambda = 0.1$, $\alpha = 0.01$.

Chapter 6

Related Work

In this chapter other and related approaches to automated diagnosis are briefly summarized and compared to the work presented in this thesis. As was discussed in the first chapter, diagnosis of machine or man is a common and important problem. Its automation would be very useful, and there have therefore been many attempts to automate it.

It is not my intention to exhaustively describe every related approach in complete detail, that would take far too much space. Instead I have chosen to look at what I think are defining and limiting features for the different systems.

Since no empirical evaluation has been done for all these systems with the same diagnosis task in mind, it is not possible to directly compare them to each other. However, they can be compared on principal grounds and on a level of comparative discussion since features and limitations are known for the approaches, *e.g.*, that uncertainties can often be better handled by a system that reasons using probability theory than a system purely based on logic.

The next section present some criteria for comparison that I have chosen to compare different methods. Then in Section 6.2 (page 94) methods from Artificial Intelligence are presented followed by Neural Networks techniques in the following section. See Section 1.3 on page 8 for a discussion on these two fields. There is also a section on other techniques.

6.1 Comparative Dimensions

To simplify the comparison I have chosen to distinguish between the following dimensions.

Symptom or Model-based approach

Most of the approaches for diagnosis have been *symptom-based*. This means that a relationship is sought from the observed symptoms to the disorders. Due to problems in relating symptoms to disorders and other representational issues, another approach called *model-based diagnosis* has emerged. The idea is to build a more or less exact model of the device. Different problems can then be simulated on this device and from that it can be concluded what the problem is with the actual device.

Specification method: Manual or Learning

The system specification can be either *manual*, or *learning*. In the manual case the programmers or system designers have to enumerate a set of rules for an expert system, or find probabilities for a Bayesian network. The learning idea is instead to find these parameters from cases in a database. There are limits to what can be done manually since full knowledge is needed. Complete knowledge about an environment can be impossible to reach in some cases.

Handling of Missing data

As discussed in Section 1.2.2 (page 6) and Section 2.1.6 on page 28, correct handling of missing data is very important for diagnosis since there will always be missing data due to the nature of the process.

Reasoning method: Logic or Probability theory

Many systems use *logic* as the main reasoning method. Other systems, like the ones in this thesis, use *probability theory*. Both can be argued for and the choice of one over the other sometimes stems from the different kinds of Artificial Intelligence systems that were discussed in Section 1.3 (page 8), symbolic and distributed systems.

Traditional and Modern systems

Traditional approaches relate to methods that have been important but are less so at the present time. For example, rule-based expert systems were in the mid-eighties the most important application for diagnosis in Artificial Intelligence. Their limitations have led to the development of Model-based diagnosis which is discussed below. The systems that are classified as *modern* are common now. Two main streams emerge, one is the model-based diagnosis systems, the other is probabilistic with neural networks in various forms.

6.1.1 Comparison charts

Table 6.1 and Table 6.2 (page 96) summarize and compares different applications in an attempt to compare them to each other. The columns in the tables compare major issues concerning automated diagnosis as discussed above.

6.2 Symbolic AI Techniques

6.2.1 Symptom-based diagnosis

Fault dictionaries

Different models of symptoms and faults can be enumerated in a *Fault Dictionary* or *Prespecified Fault Model* so that a disorder can be recognized when seen again. If there are many possible disorders, this can be hard to do. Simple problems can of course be solved like this and they can be effective. Examples of Fault Dictionaries are often found in the back of an *Owner's Manual* to home electronics, *e.g.*, a VCR. The system must be well understood and there must be a limited set of disorders with few and simple causes for this technique to work. This means that it will not be particularly effective for, *e.g.*, medical diagnosis.

Case-based reasoning

The idea behind *Case Based Reasoning* is to base it on experience. It refers to methods in Artificial Intelligence where the cases are explicitly stored. The stored case that is most similar to the current situation is then examined and whatever method that was applied to that one is used for the new case, since similar or same cases probably will benefit from the same treatment. It is also a psychological theory for human cognition, memory, planning and problem solving [Slade, 1991].

Case Based Reasoning is in a way what is done in this thesis. It may even be said that it is driven to its extreme: It is not the cases themselves but their statistical relationships that are important and that are used. Mixture

Symbolic AI techniques					
Model	**Specification**	**Missing data**	**Reasoning method**	**Traditional/ modern**	**Main limitation**
Fault dictionary	manual	no	-	traditional	For simple problems only
Case bases reasoning	manual	no	logic	traditional	Generalization?
Decision tree	learning	no	logic	traditional	Missing or initial data in traversal order
Rule-based system	manual	no	logic	traditional	Hard to build, Rule-base inconsistencies
Model-based diagnosis	manual	no(/yes)	logic	modern	Accurate model needed

Table 6.1: Comparison chart: Symbolic AI methods.

A comparison between different major symbolic AI approaches to the diagnosis problem. See section Section 6.1 on page 92 for an explanation to the column headings. Each row is a specific approach more elaborated in the text in Section 6.2.

Neural network and probabilistic techniques					
Model	**Specification**	**Missing data**	**Reasoning method**	**Traditional/ modern**	**Main limitation**
Feed-forward neural network	learning	no	probability theory	modern	All inputs must be known
Simple Bayes system	learning	no	probability theory	traditional	Independence assumptions not valid
Mixture model system	learning	yes	probability theory	modern	Large data sets needed
Growing networks	learning	no	probability theory	traditional	Intractable for non-trivial systems
Probabilistic network	manual (some learning)	yes	probability theory	modern	Probabilities must be estimated

Table 6.2: Comparison chart: Neural network and probabilistic methods. A comparison between different major neural network and probabilistic approaches to the diagnosis problem. See section Section 6.1 on page 92 for an explanation to the column headings. Each row is a specific approach more elaborated in the text in Section 6.3 (page 101).

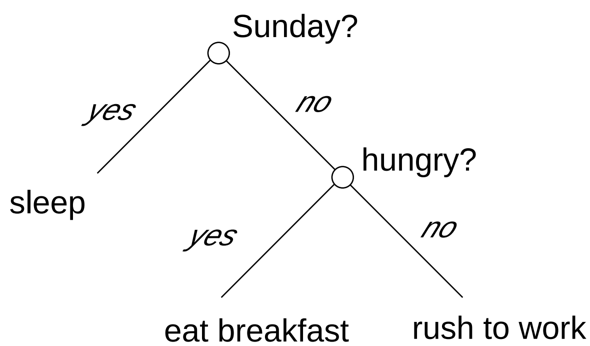

Figure 6.1: An example of a small decision tree for a day in the life.

modeling can also be seen as a clustering method. In fact, one of the most popular such methods, k-means clustering, was used to find good initial values for the parameters before the EM-algorithm was run. The resulting clusters will be prototypical cases.

Decision trees

Another approach to the diagnosis problem is to construct decision trees by inductive methods [Quinlan, 1986]. In this case a decision tree is constructed in which nodes correspond to attributes, branches to values of attributes, and leaves to outcomes, see Figure 6.1. This approach will give an optimal solution to the query-reply problem based on the training set. There are however some drawbacks.

Questions have to be asked in a certain order since the decision tree is statically constructed before the problem solving begins. To use the decision tree, the system must start from the root of the tree and move downwards from there by choosing branches. Possible initial information cannot be effectively used unless this information is what is needed in the traversal. The questions always have to be answered in the same order. Contradictions in user input may also confuse the system by leading it into a wrong branch from which it cannot find its way out of without back-tracking. Missing data and noise in the training set can also give rise to problems [Sharpe and

Solly, 1995].

Rule-based systems

Diagnosis systems have been successfully built using rule-based production systems [Davis and Lenat, 1982] and perhaps the most well-known examples are such. If-then rules are specified and there is an inferential engine that applies the rules to observed evidence and the results of other such rule applications. The rules and the data are stored in a knowledge base.

When a system is constructed the problem-solver's knowledge is condensed into rules and facts. This task can be straight-forward for a smaller system, but will be more complicated otherwise.

Up to the mid-eighties, diagnosis was one of the largest application domains of Artificial Intelligence [Milne, 1987]. The *INTERNIST* system for internal medicine [Miller *et al.*, 1982], later renamed *QMR* for *Quick Medical Reference* is a good example of a successful Expert System. *MYCIN* for blood infections [Shortliffe, 1976] is another example.

Experience has shown that rule-bases can be hard and time consuming to construct due to the need for manual specification. Inconsistencies between different applicable rules and derived facts may also arise when new rules are added to an existing database if it is large. The reason for this is that it can be hard to predict the consequences of an interdependent set of rules. Inconsistencies may also be accidentally hidden in the rules since each rule has to be explicitly stated. There is also a strong domain dependence. Knowledge bases can often not be reused for new applications [Crow and Rushby, 1994].

Attempts have been made to construct rules automatically, for example by using information theoretic reasoning [Goodman *et al.*, 1989] where rules are constructed to contain as much information as possible. There seem to be a limited applicability to this since it based on machine learning and the fact that there are much more interesting ways to do this. A benefit of this approach though, is that the resulting rules can be quite easy to understand compared to many machine learning techniques which can be much harder to understand.

Another example of learning in this respect is [Pazzani, 1987], where failure of the system is used to derive new additional rules. Failures are characterized as an inconsistent prediction, a violation of rule use or an unexpected input. For each of these there is a strategy to modify the rule base.

Work on rule-based systems can be seen to have evolved into model-based diagnosis with the realization of some of its limitations.

6.2.2 Model-based diagnosis

The basic idea in model-based diagnosis is to use structural and behavioral models of an object for the diagnosis. Most modern work on diagnosis in Artificial Intelligence is based on this approach. This type of diagnosis is often called "reasoning from first principles" since general causal principles are applied to the models.

Model-based diagnosis works by building an explicit model of the domain. The model is used to predict the future behaviour of an object, which is compared to the behaviour of the actual device. It is assumed that the model is perfect so that any differences or discrepancies are the result of faulty components.

A formalization in logic was done by [Reiter, 1987] in his *Theory of Diagnosis from First Principles*. Often the diagnosis model comes from the actual design model of the machine [Genesereth, 1984]. These techniques have been successfully used for fault diagnosis of, *e.g.*, electronic circuits. There are reviews of model-based techniques in [Milne, 1987; Davis and Hamscher, 1988; Crow and Rushby, 1994].

A number of self-explanatory tasks for model-based diagnosis have been identified: hypothesis generation, hypothesis testing and hypothesis discrimination [Davis and Hamscher, 1988; Crow and Rushby, 1994]. Other analyses have additional tasks since they also count the modeling in itself, and prediction using the model as separate tasks. The following tasks are performed:

Modeling. The model can be more or less precise so that it is an exact model or an approximation. Often hierarchical models with the same structure as the object are desirable.

Prediction. In this phase the behaviour of the model is explored. This can be done in many ways and is often an expensive part in time and resources. Some examples of prediction methods are: event-driven simulation, stepwise numeric simulation and qualitative reasoning.

Candidate generation. This task is a search for likely candidates. It can be, *e.g.*, upstream tracing which means that faulty components are probably located upstream to a point that is found to be discrepant—if the system is viewed as a process. Other approaches are also possible, see [Crow and Rushby, 1994].

Candidate testing. In this step different faults are simulated in the candidate components created in the previous step.

Candidate discrimination. A search for the correct faulty component is needed when several candidates are found. There are several common strategies for this too.

Model-based diagnosis can be a very good approach to diagnosis. The diagnosis will be perfect when an exact model is available. An additional advantage is that the resulting explanation is clear and easy to understand since the functioning is intuitive. It is how we by instinct would diagnose a simple device. It may be the natural way of reasoning for a automobile mechanic, but whether a doctor reasons like this I think is open for discussion.

The reason for this is that it can be hard to construct an exact model of a complex system. When there is no known model, which is the case in, *e.g.*, some areas of medicine, it can thus be hard to use this approach. For this reason I have deliberately chosen to use medical diagnosis in my examples since a model-based approach would in general be quite complicated due to the fact that it is not known in enough detail how many things in the body function and communicate. This line of work has however been pursued for medical diagnosis, with mechanical models of the heart [Larsson, 1995] for which a "plumbing" model can be constructed. It is not clear whether this extends beyond such models.

There are also examples of hybrid systems which combine model-based diagnosis with one or several symptom-based approaches.

6.3 Neural Network Techniques

6.3.1 Machine Learning

Feed-forward neural networks

Multi-layer neural networks have been successfully applied to diagnosis problems [Baxt, 1990; Kononenko, 1989; Shavlik *et al.*, 1991; Goodman *et al.*, 1992; Saito and Nakano, 1988; Sharpe and Solly, 1995; Burke *et al.*, 1995], and can be used as discussed in Section 1.2 (page 4). In fact, most feed-forward systems estimate posterior probabilities with a suitable encoding of the output units [Richard and Lippmann, 1991].

The main drawback is that they need full information about a case, *i.e.*, the values of all of the variables at once. This is something that rarely if ever is available in a real situation. Moreover, they cannot handle missing data, neither during learning nor during the classification. The approaches that have been used for this are intractable for anything but small problems, see Section 2.1.6 on page 28. Success or failure also often depends on the input representation and system architecture, which can make the approach more or less domain independent.

Adaptive automated diagnosis

This thesis presents two systems that can learn by machine learning from a set of cases. They can handle missing data since they model the joint probability distribution of the data. The first system called Simple Bayes rely on an approximation that makes it useful only for simple systems. It is presented in Chapter 3 (page 42). The second system use mixture models and decision theory so that it can handle complex data sets. It is presented in Chapter 4 on page 60.

Growing networks

One way to overcome the limitations of single-layer neural nets (or equivalently linear models) is to add hidden layers as discussed in the previous chapter. It is sometimes possible to create hidden layer units one by one based on some criteria that will minimize the error for the training set. Such systems are called growing networks [Fahlman and Lebiere, 1990; Frean, 1990; Mézard and Nadal, 1989].

In the case of Simple Bayes models (Chapter 3 (page 42)) the criteria can be the correlation between (input) units. These correlations can be found by searching. Unfortunately this search has to go through the entire exponential combination space in the worst case. This is of course no simple matter if the problem is of an interesting dimensionality. This has been pursued empirically by [Ekeberg and Lansner, 1988; Holst and Lansner, 1993]. For larger than small problems the search will be unfortunately be intractable.

[Weigend and Gershenfeld, 1992] discuss this problem and come to the conclusion that it is usually beneficial to use one of the standard parametric neural network learning techniques for the search for parameters instead of doing an exhaustive manual search.

6.3.2 Manual specification

Probabilistic networks

A general and exact model of a joint probability distribution (Section 2.1.5 on page 27) can be found by finding out what variables are independent, and then use that information to construct a hierarchical model by the use of the chain rule in probability theory.

The resulting models are called *Bayesian networks* or *Probabilistic networks* [Pearl, 1988; Henrion *et al.*, 1991; Russell and Norvig, 1995]. Causal dependencies can be used to find independencies and detailed domain knowledge is often needed for this reason.

Manual construction is necessary for a complex systems. In the Pathfinder system for lymph node pathology [Heckerman *et al.*, 1992; Heckerman and Nathwani, 1992] about 14,000 conditional probabilities had to be assessed by an expert pathologist.

They also had to divide the problem into independent partitions, called probabilistic similarity networks [Heckerman, 1991], to make it more manageable. This allowed them to assess only a fraction of the 75,000 original probabilities, but what remains is still of considerable size. It is inevitable that errors will occur when such large numbers of assessments are involved, but it may also be the case that exact values may not always be necessary.

Although the models are very powerful when built, there are presently no general machine learning methods for their construction which would remove the problem with manual construction. Recent work has shown that certain subtypes of probabilistic networks can be learned [Neal, 1992; Russell *et al.*, 1994; Heckerman, 1995; Saul and Jordan, 1994; Saul *et al.*, 1995].

An additional drawback is that general probabilistic inference on Bayesian networks is NP-hard, even for restricted networks. Learning has also been shown to be NP-hard [Chickering *et al.*, 1994], see also [Heckerman, 1995]. These results are for exact learning.

There are also stochastic simulation methods but they can—as is the case all Monte Carlo methods [Press *et al.*, 1992]—be very inefficient since a large number of iterative steps are required. [Myllymäki, 1993] presents a system that can evaluate certain kinds of Bayesian networks but learning still has to be done manually.

6.4 Other techniques

6.4.1 Hybrid systems

Hybrid systems are systems built out of different parts, where each part has shown to be good at solving one aspect of a problem. It can be a combinations of rule based systems, artificial neural networks and traditional computer science methods.

Standard symbolic Artificial Intelligence techniques be used to do high-level processing while artificial neural networks provide basic low-level robustness. See, *e.g.*, [Hinton, 1990; Kozato and de Wilde, 1991; Saito and Nakano, 1988; Caudill, 1991].

Expert systems that have connectionist networks for their knowledge bases are also discussed in [Gallant, 1988], where an artificial neural network is used to build a system that propagates information by logical inference. Network units communicate the discrete values true, false and unknown. This work is therefore more like an inference machine than an artificial neural network.

There are also hybrid combinations of model-based and symptom-based techniques.

6.4.2 Knowledge Discovery in Databases

Knowledge Discovery in Databases, also known as *Data Mining*, is a relatively new field that has attracted attention. The idea is that there are enormous amounts of data in various databases after a couple of decades of computerization. The problem is how to extract relevant information from the data.

Data mining is a computer-assisted process of selecting and analyzing the data to find meaning in it. The goal is to identify novel and understandable patterns in the data. It is an umbrella for several previous methods, *e.g.*, clustering, automatic finding of classification rules, or anomaly detection. It can also be said to be a subfield of machine learning that is concerned with extracting understandable knowledge from large data sets [Piatetsky-Shapiro, 1995].

Much of the present work is aimed at business data, *e.g.*, point-of-sale transaction collected by a chain of retail stores. The extracted information could be used in, e.g., marketing [Hedberg, 1995].

Chapter 7

Summary and Conclusions

This chapter summarizes the thesis, its contributions, and provides some pointers for further work.

7.1 Summary of the Thesis

Diagnosis of machine failure or disease are common everyday tasks that are performed daily by many experts in different professions. Its automation would yield many advantages, not the least monetary for machine diagnosis, but also in terms of reduced suffering or pain for a patient in medical treatment.

Diagnosis has however proven very hard to automate in a robust way. One of the main reasons is that an expert often is not aware of how the diagnosis is performed. It can therefore be hard to attain a formal or algorithmic description, which is the base for traditional approaches, *e.g.*, rule-based expert systems or model-based diagnosis systems.

This thesis presents a complete system based on probability and decision theory. It has the ability to learn from a case database, thereby overcoming the above specification problem. The model is discovered from a database instead of having to be specified by the system designer.

Missing data, both at the time of diagnosis and in the training set, is always present in diagnosis due to the nature of the diagnostic process. The joint probability distribution of the data is therefore modeled. This results in a principled solution that will give the correct result in any situation.

Mixture models were used to model the joint probability distributions. This resulted in an efficiently learnable model that was very tractable even for large systems. The parameters were determined by the Expectation-Maximization algorithm, which can also handle missing data in the training set, something that is very common in real-world databases.

Decision theory was used to find the most informative next question to ask at any time in order to minimize the total cost for the diagnosis and to minimize the consequences of misdiagnosis.

Temporal-difference reinforcement learning was used to automatically find good utility values by using an expert's assessments of diagnosis sequences created by the system as a reinforcement signal. This also allows for the use of non-linear utility functions instead of the customary linear functions.

It is demonstrated that the systems described do work well with quantitative and qualitative examples. Several small databases and a medical data set for heart disease were used.

7.2 General Discussion

7.2.1 Contributions

This thesis has contributed with the following:

- The presentation of a complete system that learns all of its parameters from a case database, thereby making the system independent of an expert's manual knowledge specification.
- The realization that diagnosis is a missing data problem, and that the joint probability distribution is necessary for a correct solution.
- There are always missing pieces of data in the databases. This has to be managed in a principled and correct way. It can be handled by the Expectation-Maximization algorithm to find parameter values.
- The use of reinforcement learning to elicit the utility values in a principled manner. This eliminates the use of an expert for the assessment of these.

7.2.2 Features

The presented diagnosis system has these features:

- The system is based on sound principles from probability and decision theory. No *ad hoc* methods or heuristics are used.

- The system is learning. Therefore there are no problems with inconsistencies in the knowledge base that can arise from manual specification.

- Any amount of initial knowledge can be used. In other systems only certain data can be used, *e.g.*, in decision trees data can only be used if it is needed during the traversal of the tree.

- The predicted probabilities of all of the outcomes are the result of a diagnosis, not just the most likely. The user can then decide on its relevance.

- The system always presents a current estimate of the probabilities for the disorder outcomes. When no information has been given, the estimate is the *a priori* probabilities for the different disorders.

- The system only recommends the next question to answer. Any question can be answered at any time.

7.2.3 Drawbacks

There are some problems with the presented approach that should be mentioned. Most of them are minor. They are discussed below.

Availability of databases

It is unfortunately at present very hard to find suitable databases since they are mostly proprietary. Future prospects for medical databases should be good. There are now hospitals that have totally abandoned paper patient journals and are instead using computerized systems. Insurance companies and Health Maintenance Organizations (HMO)s may find that it is in their interest to develop systems of the proposed type to save money and to get more reliable diagnoses. It should also be fairly easy to generate data for machine diagnosis.

Common knowledge

One of the most attractive features of the present approach is the inherent domain independence that arises from the fact that the system learns by itself. There is a negative consequence of this, and that is that the system is lacking the common background knowledge that we all share. It would be useful if there was a way to encode this in the described systems. See also the discussions on shallow and deep knowledge representation and on learning machines in Section 1.3.1 (page 8).

Explanatory Capability

One important aspect of a diagnosis system is its ability to explain why it reached a certain conclusion. This factor is important for user acceptance of an automated diagnosis system. Since the mixture component basis functions have local support and since we have estimates of the probability of each function having generated the observed data, an explanation for the conclusion can be found. Something similar to this has been used by [Tresp *et al.*, 1994] in a different but related context.

It might also be worth pointing out that we everyday trust, more or less, other things that we do not fully understand, with the *brain* being the most interesting example. Thus, we can perhaps never expect to fully trust any diagnostic expert at all. We should try to define how we trust something or somebody else, be it human or machine, and study what makes us trust a physician or any other expert.

Only one disease at a time can be handled

Following the derivation of the EM-algorithm, there can only be one disease per case in the data set. This limitation is discussed in Chapter 4 on page 60.

7.3 Further Work

Several properties of these models remain to be investigated:

Reinforcement learning

An alternative to learning the utility or value function in Chapter 5 (page 80) is to directly learn the optimal actions to take at each state. A method for this is Q-learning [Watkins, 1989]. In our case this would be to learn which question to ask in each situation instead of the utility values. The myopic approximation would then not be necessary, but it would lead us away from the normal decision theoretic formulation and would therefore be less attractive.

In this regard it would also be interesting to explore methods to automatically find good values for the α and λ parameters for the temporal-difference algorithm, perhaps along the lines of [Sutton and Singh, 1994], to avoid the search for parameters of the utility model in Chapter 5 on page 80.

Non-linear utilities

The work presented in Chapter 5 (page 80), where reinforcement learning was used to find utility values, allows for an extension to non-linear utility models. Such models would be very complicated to specify manually since there is no direct interpretation for each of the values. With the use of the Temporal-Difference reinforcement learning algorithm it is possible to do this in the same way that was done with linear utilities.

Automatically find the optimal number of mixture components

There is no way to automatically find the optimal number of mixture components for the mixture models [McLachlan and Basford, 1988]. By using Minimum Description Length arguments, it should be possible to do this with optimization.

Alternative question selection

An alternative way to choose a new question is to evaluate what the variance change in the output variables will be when a variable is changed from missing to observed. The idea is that a variable known with certainty has zero variance [Chávez and Henrion, 1994]. Therefore, the variable with the largest *conditional variance* should be selected as the query.

A similar technique is the concept of *Optimal Experiment Design* [Fedorov, 1972], where experiments are designed to minimize the variance of a parameterized model, thereby maximizing the confidence in the model. This has been introduced into the neural network domain for other purposes by [Cohn, 1994; MacKay, 1992].

Dynamical systems

In the present work it is assumed that the objects do not change while the diagnosis is being done due to the static model of the joint probability densities. By using Hidden Markov Models [Rabiner and Juang, 1986], which essentially are time-expanded mixture models. A similar algorithm to the EM-algorithm is then used.

More training data would be needed in this case, and a choice of what kind of structure is also needed, just like there has to be a way to find how many mixture components are necessary in the present models as discussed above. [Forbes *et al.*, 1995] discuss how dynamic belief networks can be constructed.

Learning in Probabilistic networks

Probabilistic or Bayesian networks (Section 6.3.2 on page 102) are much more general hierarchical models than mixture models. They can, e.g., handle multiple diseases which the presented models cannot. Most probabilistic networks are hand-crafted but there are at this time some learning methods [Neal, 1992; Russell *et al.*, 1994; Heckerman, 1995; Saul and Jordan, 1994; Saul *et al.*, 1995].

None of these are however applicable for general systems. Learning in these kinds of models is quite new and future work will probably find learning algorithms for some of them. Mixture models can in fact be seen as flat Bayesian networks. Hierarchical models would remove the requirement for only one disease per case in the database.

Appendix A

Databases

This appendix describes the databases that were used when developing and evaluating the systems described in the chapters of the thesis.

Since machine learning is used no background knowledge is necessary. However, to enable evaluation by a non-expert, I chose to use familiar test databases for the work in this thesis. To find which, *e.g.*, animal is present given some observations and to find which disease is present given symptoms and other information is the same thing. This is further discussed in Chapter 1 (page 2).

A.1 A medical database

To demonstrate the capabilities of the diagnosis systems in Chapter 4 on page 60 and Chapter 5 (page 80), a real-world data base, the Cleveland heart disease data set from the University of California, Irvine Repository of Machine Learning Databases [Murphy and Aha] has been used.

The data set consists of 303 examples of four types of heart disease and its absence. There are thirteen continuous or nominally valued variables as shown in Table A.1. There are missing data in the database.

A.2 Small test databases

The databases listed below were used to test the Simple Bayes system in Chapter 3 (page 42). To show that the mixture model system described in Chapter 4 on page 60 performs better, they were also used there.

These databases are further explained and fully reproduced in [Stensmo, 1991].

Animals and Animals2. The database *Animals* consists of 32 animal descriptions each with 10–15 observations. To stress that an animal is different from the prototype, *e.g.*, that a `penguin` does not fly, special units have been used for this. *Animals2* has fewer observations than *Animals*. It is more compressed and does not contain the above mentioned extra observations. *Animals2* does however not show any

Table A.1: The Cleveland Heart Disease database.

	Observation	**Description**	**Values**
1	age	Age in years	continuous
2	sex	Sex of subject	male/female
3	cp	Chest pain	four types
4	trestbps	Resting blood pressure	continuous
5	chol	Serum cholesterol	continuous
6	fbs	Fasting blood sugar	$<$, or $>$ 120 mg/dl
7	restecg	Resting electrocardiographic result	five values
8	thalach	Maximum heart rate achieved	continuous
9	exang	Exercise induced angina	yes/no
10	oldpeak	ST depression induced by exercise relative to rest	continuous
11	slope	Slope of peak exercise ST segment	up/flat/down
12	ca	Number major vessels colored by flouroscopy	0-3
13	thal	Defect type	normal/fixed/ reversible
	Disorder	**Description**	**Values**
14	num	Heart disease	Not present/ four types

significantly different behaviour in testing, so it seems that these extra units are not needed.

Mushroom. The *Mushroom* database contains descriptions of 28 different types of mushroom, each described by about 10 observations.

Bumble-bees. *Bumble-bees* consists of two examples each of 24 bumble-bee species, one for the queen and one for the male. It consists of 48 patterns of which 24 has the observation `queen` and *vice versa*.

Tree. *Tree* is a true hierarchical binary tree with 39 leaves or outcomes. The maximal tree depth is 3. It is deliberately made very unbalanced.

Random2 and Random3. These databases are purely random and are designed to be opposites of *Tree*, *i.e.*, not at all hierarchical. *Random2* has 26 12-observation patterns where each observation is an integer chosen from the range $[0, 74]$. The database *Random3* has 23 patterns in groups of three: 10 patterns with integer observations in the range $[0, 44]$, 10 from $[30, 74]$, and 3 from $[0, 74]$.

List of Figures

List of Tables

Bibliography

Ackley, D. H., G. E. Hinton and T. J. Sejnowski (1985). A learning algorithm for Boltzmann machines. *Cognitive Science*, **9**. Reprinted in [Anderson and Rosenfeld, 1988].

Ahmad, S. and V. Tresp (1993). Some solutions to the missing feature problem in vision. In *Advances in Neural Information Processing Systems*, volume 5, pp 393–400. Morgan Kaufmann, San Mateo, CA.

Allen, Arnold O. (1978). *Probability, Statistics, and Queueing Theory*. Academic Press, Orlando, FL.

Anderson, J. A. and E. Rosenfeld, editors (1988). *Neurocomputing: Foundations of Research*. MIT Press, Cambridge, MA.

Barto, A. G., R. S. Sutton and C. W. Anderson (1983). Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, **13**. Reprinted in [Anderson and Rosenfeld, 1988].

Baxt, W. G. (1990). Use of an artificial neural network for data analysis in clinical decision-making: The diagnosis of acute coronary occlusion. *Neural Computation*, **2(4)**, 480–489.

Bayes, Thomas (1763). An essay toward solving a problem in the doctrine of chances. *Philosophical Transactions of the Royal Society, London*, pp 370–418. Reprinted in [Deming, 1940].

Broomhead, D. S. and D. Lowe (1988). Multivariable functional interpolation and adaptive networks. *Complex Systems*, **2**, 321–355.

Burke, H. B., D. B. Rosen and P. H. Goodman (1995). Comparing the prediction accuracy of artificial neural networks and other statistical models for breast cancer survival. In *Advances in Neural Information Processing Systems*, volume 7, pp 1063–1067. MIT Press, Cambridge, MA.

Caudill, Maureen (1991). Expert networks. *BYTE*, pp 108–116, October.

Chávez, Tom and Max Henrion (1994). Efficient estimation of the value of information in Monte Carlo models. In Lopez de Mantaras, R. and D. Poole, editors, *Tenth Conference on Uncertainty in Artificial Intelligence*, Seattle, WA.

Chickering, David M., Dan Geiger and David Heckerman (1994). Learning Bayesian networks is NP-hard. Technical Report MSR-TR-94-17, Microsoft Research, Redmond, WA.

Cohen, L. Jonathan (1981). Can human irrationality be experimentally demonstrated. *The Behavioral and Brain Sciences*, **4**, 317–370. With commentary.

Cohn, David A. (1994). Neural network exploration using optimal experiment design. In *Advances in Neural Information Processing Systems*, volume 6, pp 679–686. Morgan Kaufmann, San Mateo, CA.

Crites, Robert H. and Andrew G. Barto (1996). Improving elevator performance using reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 8. MIT Press, Cambridge, MA.

Crow, Judy and John Rushby, (1994). Model-based reconfiguration: Diagnosis and recovery. NASA Contractor Report 4596, Computer Science Lab, SRI International, Menlo Park, CA.

Davis, R. and W. Hamscher (1988). Model-based reasoning: Troubleshooting. In Shrobe, H. E., editor, *Exploring Artificial Intelligence*, pp 297–346. Morgan Kaufmann, San Mateo, CA.

Davis, Randall and Douglas B. Lenat (1982). *Knowledge-Based Systems in Artificial Intelligence.* McGraw-Hill, New York, NY.

Dayan, P. (1992). The convergence of TD(λ) for general λ. *Machine Learning*, **8**, 341–362.

Deming, W. Edwards, editor (1940). *Facsimiles of two papers by Bayes: I. An essay toward solving a problem in the doctrine of chances, with*

Richard Price's foreword and discussion. Phil. Trans. Royal Soc., pp.370-418, 1763. With a commentary by Edward C. Molina. II. A letter on asymptotic series from Bayes to John Canton; pp.269-271 of the same volume. With a commentary by W. Edwards Deming. The Graduate School, Department of Agriculture, Washington, DC. See [Bayes, 1763].

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

Dobson, Annette J. (1990). *An Introduction to Generalized Linear Models.* Chapman and Hall, London, UK, second edition.

Duda, R. O. and P. E. Hart (1973). *Pattern Classification and Scene Analysis.* John Wiley & Sons, Inc., New York, NY.

Eddy, David M. and Charles H. Clanton (1982). The Art of Diagnosis: Solving the clinicopathological exercise. *New England Journal of Medicine*, **306(21)**, 1263–1268.

Ekeberg, Örjan and Anders Lansner (1988). Automatic generation of internal representations in a probabilistic artificial neural network. In Personnaz, L. and G. Dreyfus, editors, *Neural Networks from Models to Applications*, pp 178–186, Paris. I.D.S.E.T. Proc. nEuro-88, the First European Conference on Neural Networks.

Encyclopædia Britannica (1992). *The New Encyclopædia Britannica.* Encyclopaedia Britannica, Inc., Chicago, Ill., 15th edition.

Fahlman, S. E. and C. Lebiere (1990). The cascade-correlation learning architecture. In *Advances in Neural Information Processing Systems*, volume 2, pp 524–532. Morgan Kaufmann, San Mateo, CA.

Fedorov, Valerii Vadimovich (1972). *Theory of Optimal Experiments.* Academic Press, New York, NY.

Forbes, Jeff, Tim Huang, Keiji Kanazawa and Stuart Russell, (1995). The BATmobile: towards a Bayesian automated taxi. IJCAI-95.

Frean, M. (1990). The upstart algorithm: A method for constructing and training feedforward neural networks. *Neural Computation*, **2**, 198–209.

Gallant, Stephen I. (1988). Connectionist expert systems. *Communications of the ACM*, **31(2)**, 152–169.

Genesereth, M. R. (1984). The use of design descriptions in automated diagnosis. *Artificial Intelligence*, **24**, 411–436.

Gennari, J. H., P. Langley and D. Fisher (1989). Models of incremental concept formation. *Artificial Intelligence*, **40**, 11–62.

Ghahramani, Z. and M. I. Jordan (1994a). Learning from incomplete data. Technical Report AI Memo No 1509, CBCI Paper No 108, AI Lab, Massachusetts Institute of Technology, Cambridge, MA.

Ghahramani, Z. and M. I. Jordan (1994b). Supervised learning from incomplete data via an EM approach. In *Advances in Neural Information Processing Systems*, volume 6, pp 120–127. Morgan Kaufmann, San Mateo, CA.

Goodman, Rodney M., John W. Miller and Padhraic Smyth (1989). An information theoretic approach to rule-based connectionist expert systems. In Touretzky, David S., editor, *Advances in Neural Information Processing I*, pp 256–263. Morgan Kaufmann, San Mateo, USA.

Goodman, R. M., P. Smyth, C. M. Higgins and J. W. Miller (1992). Rule-based neural networks for classification and probability estimation. *Neural Computation*, **4(6)**, 781–804.

Gorry, G. Anthony and G. Octo Barnett (1967). Experience with a model of sequential diagnosis. *Computers and Biomedical Research*, **1**, 490–507.

Grayson, C. J. (1960). Decisions under uncertainty: Drilling decision by oil and gas operators. Technical report, Harvard Business School, Boston, MA.

Grimmett, G. R. and D. R. Stirzaker (1992). *Probability and Random Processes*. Oxford Science Publications, Oxford, UK, second edition.

Guha, R. V. and D. B. Lenat (1990). Cyc: a midterm report. *AI Magazine*, **11(3)**, 32–59.

Haykin, Simon (1994). *Neural Networks: A Comprehensive Foundation*. Macmillan, New York, NY.

Hebb, D. O. (1949). *The Organization of Behavior*. Wiley, New York. Partially reprinted in [Anderson and Rosenfeld, 1988].

Heckerman, David E. (1991). *Probabilistic Similarity Networks*. MIT Press, Cambridge, MA.

Heckerman, David (1995). A tutorial on learning Bayesian networks. Technical Report MSR-TR-95-06, Microsoft Research, Redmond, WA.

Heckerman, D. E. and B. N. Nathwani (1992). Toward normative expert systems: Part II. Probability-based representations for efficient knowledge acquisition and inference. *Methods of Information in Medicine*, **31**, 106–116.

Heckerman, D. E., E. J. Horvitz and B. N. Nathwani (1992). Toward normative expert systems: Part I. The Pathfinder project. *Methods of Information in Medicine*, **31**, 90–105.

Hedberg, Sara Reese (1995). The data gold rush. *BYTE*, October.

Henrion, M., J. S. Breese and E. J. Horvitz (1991). Decision analysis and expert systems. *AI Magazine*, **12(4)**.

Hertz, John, Anders Krogh and Richard Palmer (1991). *Introduction to the Theory of Neural Computation*. Addison-Wesley.

Hinton, G. E. (1990). Preface to the special issue on connectionist symbol processing. *Artificial Intelligence*, **46(1–2)**, 1–4.

Hinton, Geoffrey E., Peter Dayan, Brendan J. Frey and Radford M. Neal (1995). The wake-sleep algorithm for unsupervised neural networks. *Science*, **268**, 1158–1161.

Holst, A. and A. Lansner (1993). A Bayesian neural network model with extensions. Technical Report TRITA-NA-P9325, Dept. of Numerical Analysis and Computing Science, Kungliga Tekniska Högskolan (Royal Institute of Technology), Stockholm, Sweden.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences, USA*, **79**. Reprinted in [Anderson and Rosenfeld, 1988].

Howard, Ron A. (1966). Information value theory. *IEEE Transactions on Systems Science and Cybernetics*, **SSC-2**, 22–26.

Howard, Ron A. (1980). On making life and death decisions. In Schwing, Richard C. and Walter A. Albers, Jr., editors, *Societal risk assessment: How safe is safe enough?* Plenum Press, New York, NY.

Howard, Ron A. (1990). From influence to relevance to knowledge. In Oliver, R. and J. Smith, editors, *Influence Diagrams, Belief Nets, and Decision Analysis*. Wiley, New York, NY.

Hush, Don R. and Bill G. Horne (1993). Progress in supervised neural networks — What's new since Lippmann? *IEEE Signal Processing Magazine*, pp 8–39, January.

Jacobs, R. A., M. I. Jordan, S. J. Nowlan and G. E. Hinton (1991). Adaptive mixtures of local experts. *Neural Computation*, **3(1)**, 79–87.

Kononenko, I. (1989). Bayesian neural networks. *Biological Cybernetics*, **61**, 361–370.

Kozato, F. and Ph. de Wilde (1991). How neural networks help rule-based problem solving. In Kohonen, T., K. Mäkisara, O. Simula and J. Kangas, editors, *Artificial Neural Networks*, volume 1, pp 465–470. North-Holland, Espoo, Finland, June.

Lansner, Anders and Örjan Ekeberg (1989). A one-layer feedback artificial neural network with a bayesian learning rule. *International Journal of Neural Systems*, **1(1)**, 77–87.

Larsson, Jan Eric (1995). Model-based monitoring and diagnosis of the human body. Technical Report KSL-95-54, Knowledge Systems Laboratory, Dept. of Computer Science, Stanford University, Stanford, CA.

Ledley, Robert S. and Lee B. Lusted (1959). Reasoning foundations of medical diagnosis. *Science*, **130(3366)**, 9–21.

Little, R. J. A. (1992). Regression with missing X's: A review. *Journal of the American Statistical Association*, **87(420)**, 1227–1237.

Little, R. J. A. and D. B. Rubin (1987). *Statistical Analysis with Missing Data*. Wiley, New York, NY.

MacKay, David J. C. (1992). Information-based objective functions for active data selection. *Neural Computation*, **4**, 590–604.

McCulloch, W. S. and W. Pitts (1943). A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, **5**. Reprinted in [Anderson and Rosenfeld, 1988].

McLachlan, Geoffrey J. and Kaye E. Basford (1988). *Mixture Models: Inference and Applications to Clustering*. Marcel Dekker, Inc., New York, NY.

Mézard, M. and J.-P. Nadal (1989). Learning in feedforward layered networks: The tiling algorithm. *Journal of Physics A*, **22**, 2191–2204.

Miller, Randolph A., Harry E. Pople and Jack D. Myers (1982). Internist-1: An experimental computer-based diagnostic consultant for general internal medicine. *New England Journal of Medicine*, **307**, 468–476.

Milne, Robert (1987). Strategies for diagnosis. *IEEE Transactions on Systems, Man, and Cybernetics*, **SMC-17(3)**, 333–339.

Minsky, M. L. and S. A. Papert (1969). *Perceptrons.* MIT Press, Cambridge, MA. There is an expanded edition [Minsky and Papert, 1988].

Minsky, M. L. and S. A. Papert (1988). *Perceptrons.* MIT Press, Cambridge, MA, expanded edition. [Minsky and Papert, 1969] is the original edition.

Moody, J. (1989). Fast learning in multi-resolution hierarchies. In *Advances in Neural Information Processing Systems*, volume 1, pp 29–39. Morgan Kaufmann, San Mateo, CA.

Moody, J. and C. Darken (1988). Learning with localized receptive fields. In *Proc. 1988 Connectionist Models Summer School*, pp 133–43. Morgan Kaufmann.

Moody, J. and C. J. Darken (1989). Fast learning in networks of locally-tuned processing units. *Neural Computation*, **1(2)**, 281–294.

Murphy, P. M. and D. W. Aha. UCI repository of machine learning databases. Dept. of Information and Computer Science, University of California, Irvine, CA. URL: ftp://ics.uci.edu/pub/machine-learning-databases.

Myllymäki, P., (1993). Bayesian reasoning by stochastic neural networks. Licentiate Thesis, Helsinki University, Finland.

Neal, R. M. (1992). Connectionists learning of belief networks. *Artificial Intelligence*, **56**, 71–113.

Nowlan, S. J. (1990). Maximum likelihood competitive learning. In *Advances in Neural Information Processing Systems*, volume 2, pp 574–582. Morgan Kaufmann, San Mateo, CA.

Nowlan, Steven J. (1991). *Soft Competitive Adaptation: Neural Network Learning Algorithms based on Fitting Statistical Mixtures*. PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA.

Ormoneit, Dirk (1993). Estimation of probability densities using neural networks. Diplomarbeit, Technischen Universität München, Germany.

Papoulis, Athanasios (1984). *Probability, Random Variables, and Stochastic Processes*. McGraw Hill, New York, NY, second edition.

Park, J. and I. W. Sandberg (1991). Universal approximation using Radial-Basis-Function networks. *Neural Computation*, **3**, 246–257.

Pazzani, M. J. (1987). Failure driven learning of fault diagnosis heuristics. *IEEE Transactions on Systems, Man, and Cybernetics*, **SMC-17(3)**, 380–394.

Pearl, Judea (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA.

Piatetsky-Shapiro, Gregory, (1995). Knowledge discovery and data mining FAQ. URL http://info.gte.com/~kdd/FAQ.html.

Poggio, T. and F. Girosi (1990). Regularization algorithms for learning that are equivalent to multilayer networks. *Science*, **247**, 978–982.

Press, William H., Saul A. Teukolsky, William T. Vetterling and Brian P. Flannery (1992). *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, Cambridge, UK, second edition.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, **1**, 81–106.

Rabiner, L. R. and B.-H. Juang (1986). An introduction to Hidden Markov Models. *IEEE ASSP Magazine*, January.

Reiter, R. (1987). A theory of diagnosis from first principles. *Artificial Intelligence*, **32**, 57–95.

Richard, M. D. and R. P. Lippmann (1991). Neural network classifiers estimate Bayesian *a posteriori* probabilities. *Neural Computation*, **3(4)**, 461–483.

Rosenblatt, F. (1962). *Principles of Neurodynamics*. Spartan, New York, NY.

Rumelhart, D. E., G. E. Hinton and R. J. Williams (1986a). Learning internal representations by error propagation. In Rumelhart, D. E. and J. L. McClelland, editors, *Parallel Distributed Processing*, volume 1, chapter 8, pp 318–362. MIT Press, Cambridge, MA. Reprinted in [Anderson and Rosenfeld, 1988].

Rumelhart, D. E., G. E. Hinton and R. J. Williams (1986b). Learning representations by back-propagating errors. *Nature*, **323**, 533–536. Reprinted in [Anderson and Rosenfeld, 1988].

Rumelhart, D. E., J. L. McClelland and the PDP Research Group (1986c). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1. MIT Press, Cambridge, MA.

Russell, Stuart J. and Peter Norvig (1995). *Artificial Intelligence: A Modern Approach*. Prentice Hall, Englewood Cliffs, NJ.

Russell, S., J. Binder and D. Koller (1994). Adaptive probabilistic networks. Technical Report TR UCB//CSD-94-824, University of California, Berkeley, Berkeley, CA.

Saito, K. and R. Nakano (1988). Medical diagnostic expert system based on [a] PDP model. In *IEEE International Conference on Artificial Neural Networks*, San Diego, CA.

Saul, Lawrence and Michael I. Jordan (1994). Learning in Boltzmann trees. *Neural Computation*, **6**, 1174–1184.

Saul, Lawrence, Tommi Jaakkola and Michael I. Jordan (1995). Mean field theory for sigmoid belief networks. Technical Report 9501, Center for Biological and Computational Learning, MIT, Cambridge, MA.

Shannon, Claude E. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, **XXVII(3)**. In [Shannon and Weaver, 1949].

Shannon, Claude E. and Warren Weaver (1949). *The Mathematical Theory of Communication*. University of Illinois Press, Urbana, IL. Ninth printing 1962.

Sharpe, P. K. and R. J. Solly (1995). Dealing with missing values in neural network-based diagnostic systems. *Neural Computation & Applications*, **3**, 73–77.

Shavlik, J. W., R. J. Mooney and G. G. Towell (1991). Symbolic and neural learning algorithms: An experimental comparison. *Machine Learning*, **6**, 111–143.

Shortliffe, E. H. (1976). *Computer-Based Medical Consultations: MYCIN*. Elsevier, New York, NY.

Slade, Stephen (1991). Case-based reasoning: A research paradigm. *AI Magazine*, pp 42–55, Spring.

Stensmo, Magnus (1991). A query-reply classification system based on an artificial neural network. Technical Report TRITA-NA-9107, Dept. of Numerical Analysis and Computing Science, Kungliga Tekniska Högskolan (Royal Institute of Technology), Stockholm, Sweden. Licentiate thesis.

Stensmo, Magnus and Terrence J. Sejnowski (1994). A mixture model diagnosis system. Technical Report INC-9401, Institute for Neural Computation, La Jolla.

Stensmo, Magnus and Terrence J. Sejnowski (1995a). A mixture model system for medical and machine diagnosis. In Tesauro, G., D. S. Touretzky and T. K. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pp 1077–1084. MIT Press, Cambridge, MA.

Stensmo, Magnus and Terrence J. Sejnowski (1995b). Using temporal-difference reinforcement learning to improve decision-theoretic utilities for diagnosis. In *Proc. of the 2nd Joint Symposium on Neural Computation, University of California, San Diego and California Institute of Technology*, pp 9–16. Institute for Neural Computation, La Jolla, CA.

Stensmo, Magnus, Björn Levin and Anders Lansner (1989). A question driven classification system using a neural network model. In *Proc. Nordic Symposium on Neural Computing*, Espoo, Finland, April.

Stensmo, Magnus, Anders Lansner and Björn Levin (1991). A query-reply system based on a recurrent Bayesian neural network. In Kohonen, T., K. Mäkisara, O. Simula and J. Kangas, editors, *Artificial Neural Networks*, volume 1, pp 459–464. North-Holland, Amsterdam, The Netherlands.

Sutton, Richard S. (1988). Learning to predict by the method of temporal differences. *Machine Learning*, **3**, 9–44.

Sutton, Richard S. and Satinder P. Singh (1994). On step-size and bias in temporal-difference learning. In *Eight Yale Workshop on Adaptive and Learning Systems*, pp 91–96, New Haven, CT. Center for Systems Science, Yale University.

Tesauro, G. (1992). Practical issues in temporal difference learning. *Machine Learning*, **8**, 257–277.

Tesauro, G. (1993). TD-Gammon, a self-teaching backgammon program, achieves master-level play. *Neural Computation*, **6**, 215–219.

Tou, Julius T. and Rafael C. Gonzales (1974). *Pattern Recognition Principles*. Addison-Wesley, London. 4th printing 1981.

Tresp, V., S. Ahmad and R. Neuneier (1994). Training neural networks with deficient data. In *Advances in Neural Information Processing Systems*, volume 6, pp 128–135. Morgan Kaufmann, San Mateo, CA.

Tresp, Volker, Ralph Neuneier and Subutai Ahmad (1995). Efficient methods for dealing with missing data in supervised learning. In *Advances in Neural Information Processing Systems*, volume 7. MIT Press, Cambridge, MA.

Tversky, A. and D. Kahneman (1974). Judgement under uncertainty: Heuristics and biases. *Science*, **185**, 1124–1131.

von Neumann, John and Oscar Morgenstern (1947). *Theory of Games and Economic Behavior*. Princeton University Press, Princeton, NJ.

Watkins, Christopher John Cornish Hellaby (1989). *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, UK.

Weigend, Andreas S. and Neil A. Gershenfeld, editors. *Time series prediction: Forecasting the future and understanding the past. Proceedings of the NATO Advanced Research Workshop on Comparative Time Series Analysis*, Reading, MA. Addison-Wesley.

Xu, L. and M. I. Jordan (1994). Theoretical and experimental studies of the EM algorithm for unsupervised learning based on finite Gaussian mixtures. Technical Report 9302, Computational Cognitive Science, Massachusetts Institute of Technology, Cambridge, MA.