Intelligent Diagnosis Systems

Karthik Balakrishnan & Vasant Honavar [1]
Artificial Intelligence Research Group
Department of Computer Science
Iowa State University
Ames, Iowa 50011-1040, U.S.A.
balakris@cs.iastate.edu, honavar@cs.iastate.edu
http://www.cs.iastate.edu/~honavar/aigroup.html

[1]This research was partially supported through grants from the John Deere Foundation and the National Science Foundation (NSF IRI-9409580) to Vasant Honavar.

Abstract

This paper examines and compares several different approaches to the design of intelligent systems for diagnosis applications. These include expert systems (or knowledge-based systems), truth (or reason) maintenance systems, case-based reasoning systems, and inductive approaches like decision trees, artificial neural networks (or connectionist systems), and statistical pattern classification systems. Each of these approaches is demonstrated through the design of a system for a simple automobile fault diagnosis task. The paper also discusses the domain characteristics and design and performance requirements that influence the choice of a specific technique (or a combination of techniques) for a given application.

Keywords: Intelligent Diagnosis, Expert Systems, Model-Based Systems Case-Based Reasoning, Neural Networks, Decision Trees, Knowledge Acquisition

1 INTRODUCTION

The last few decades have seen a proliferation of intelligent systems for diagnosis, advising, and related applications (Dean *et al.*, 1995; Durkin, 1994; Ginsberg, 1993; Luger & Stubblefield, 1993; Puppe, 1993; Rich & Knight, 1991; Russell & Norvig, 1995; Stefik, 1995; Tanimoto, 1995; Winston, 1992). Intelligent systems for diagnosis have been used in a variety of domains: plant disease diagnosis, crop management problem diagnosis, credit evaluation and authorization, financial evaluation, identification of software and hardware problems and integrated circuit failures, troubleshooting of electrical, mechanical and electronic equipment, medical diagnosis, fault-detection in nuclear power systems, oil exploration, prospecting, seismic studies, etc. Despite the great variety of approaches and technologies used in the design of such systems, they all address the same pattern classification problem: the task of assigning a given input (e.g., a pattern, image, set of observations, etc.) to some category (or class). Diagnosis systems classify the observed symptoms as being caused by some specific problem (diagnosis class) while advising systems perform such a classification and suggest corrective remedies. Although this paper focuses on intelligent systems for diagnosis applications, most of the general principles and observations from the design of such systems apply to other tasks that involve pattern classification as well.

We begin with some definitions of *diagnosis* from the Webster's Dictionary. Diagnosis is defined as:

1. the act or process of deciding the nature of a disease or a problem by examination of the symptoms
2. a careful examination and analysis of the facts in an attempt to explain or understand something [a *diagnosis* of the economy]
3. a decision or opinion based on such an examination
4. a short scientific description for taxonomic classification

In short, a *diagnosis system* is one that is capable of identifying the nature of a problem by examining the observed symptoms. The output of the system is a diagnosis (and possibly an explanation or justification of the same). In many applications of interest, it is desirable for the system to not only identify the possible causes of the problem, but also propose suitable remedies. Such systems are said to be capable of *advising*.

Typically a diagnosis system is provided a set of *symptoms* (observations or measurements encoded in some machine-readable form) as input. The system's task is to identify a probable *cause* that could explain the observed symptoms. In order to perform this function, a diagnosis system obviously needs adequate *knowledge* of the domain. The necessary knowledge may be engineered into the system by its designers or the system may be endowed with the ability to acquire the necessary domain knowledge on its own, through experience. The former approach, referred to as *knowledge engineering*, is typically used in the construction of the diagnostic rules for expert systems or knowledge-based systems. The latter approach relies on the use of *machine learning* techniques for automated knowledge acquisition. Artificial Intelligence (AI) offers a broad spectrum of approaches to intelligent diagnosis, each with its own strengths and weaknesses, depending (among other things) on the nature of the diagnosis task, and the type and amount of domain knowledge that is available. This paper examines and compares various approaches to the design of intelligent diagnosis systems, emphasising their strengths and limitations. This analysis also suggests interesting possibilities for hybrid techniques that combine different approaches fruitfully.

The rest of the paper is organized as follows: section 2 develops a simple example of a diagnosis problem for the electrical system of an automobile. The purpose of this example is two-fold. First, it helps clarify the different elements that go into the formulation of a diagnosis problem. Second, the same example is used to illustrate several different approaches to diagnosis. Section 3 gives a brief overview of the different approaches to diagnosis. Sections 4 through 6 examine expert systems, model-based systems, and case-based approaches to diagnosis. Section 7 presents an overview of inductive approaches while sections 8 through 10 present decision trees, neural networks, and statistical approaches in the context of diagnosis. Section 11 concludes with a summary and discussion of interesting research directions for building hybrid systems.

2 A SAMPLE DIAGNOSIS TASK

Consider a company that manufactures automobiles. In order to provide better customer service and support, let us assume that the company has set up telephone help lines to diagnose automobile malfunctions experienced by their customers. For example, the company might expect to receive a call from a customer complaining that his engine does not start. The company expects to help the user by diagnosing the cause of the problem. Suppose the call is answered by a human expert (adept at diagnosing such automobile problems), who knows through experience or knowledge that automobile engines will not start if either the ignition, or the battery, or both are faulty. (Note that we ignore other possible causes like faulty spark-plugs, carburetor, fuel injector, etc., to keep the discussion simple.) To pinpoint the exact source of the problem, the expert might use her knowledge that if the battery is faulty the headlights won't work either. Then she might ask the user to try turning on his headlights. If she is informed that they work, by simple elimination she would diagnose the problem as faulty ignition, and if they don't, she would suggest that the user might have a faulty battery.

Since human expertise is expensive and often prone to errors (e.g., due to stress, fatigue, etc.) the company might be interested in an intelligent diagnosis system that could play the role of the human expert. In order to be useful in practice, such a system would have to be capable of diagnosing faults with accuracy comparable to that of a human expert under identical conditions. Let us now formulate a sample diagnosis problem for automobile electrical system failures. To keep the problem simple, let us assume that we are only concerned with the possible malfunctions or failures of the following components:

- Battery
- Bulbs (headlight)
- Wiper Motor
- Ignition

In other words, the system has to identify problems arising from a failure of one or more of these components. For example, headlights will not work if the bulbs fail. Hence, if we observe that the headlights don't work, we might diagnose that the bulbs are faulty. To keep the example simple, we further assume that the relevant observable events or symptoms are restricted to the following:

- Headlights work/don't work
- Engine starts/doesn't start
- Wipers work/don't work

We should point out that this specification of the problem assumes that the only causes of the symptoms are the batteries, bulbs, wiper motor, and ignition. For example, if the engine does not start we assume that the fault lies either with the battery or the ignition only; we ignore other possible causes like the improper injection of the fuel-air mixture, worn-out spark plugs, etc. We also assume that the wipers and headlights are tested without starting the engine (i.e., they are powered by the battery). This scenario, albeit contrived, simplifies the discussion that follows. For simplicity, we also assume that *both* the headlight bulbs fail, if at all. Given the observed symptoms, the diagnosis system then has the task of ascribing the symptoms to the failure of one (or more) of the components. In passing, let us stress that this diagnosis task is overly simplified: real-world problems can involve hundreds of observable symptoms involving a large number of components. However, as we shall see, the general principles remain the same.

3 DIAGNOSIS SYSTEMS: AN OVERVIEW OF APPROACHES

Our objective is to build a system that produces accurate diagnosis efficiently. How will the system come to know of the relationship between the observed symptoms and the consequent diagnosis? How will the system represent this relationship? How will it use this representation for diagnosing faults? The different approaches to diagnosis can be distinguished based on how these questions are addressed. Consider a scenario in which a *model* of the domain is available. Such a model explicitly represents the structure of the system, i.e., its constituent components and their organization (Mozetic, 1992). A diagnosis problem arises when the system's observed behavior conflicts with the system's expected behavior, and the task is to identify the (faulty) system component(s) that explain the anomaly. In our diagnosis example, since the automobile was designed and constructed by a group of engineers, we have available a more or less exact model of the functioning of the automobile. For example, the automobile was designed and built with an engine that could be started by turning the ignition. For the engine to start up, fuel-air mixture must be injected into the combustion chamber and the ignition must be cranked thereby allowing an electrical current to flow from the battery to the spark-plugs. The spark-plug produces a spark, igniting the air-fuel mixture and starting the engine, etc. Now if we have this model of the automobile available to us and the engine fails to start, we can easily trace the problem to either a faulty ignition system or a faulty battery (assuming the problem is localized to the electrical system). Based on further information (e.g., concerning the functioning of the headlights), one could possibly diagnose the exact cause. This is the principle behind *model-based diagnosis systems* (Durkin, 1994; Mozetic, 1992; Puppe, 1993). *Reason maintenance systems* (Forbus & de Kleer, 1993) can be used in model-based diagnosis and one such approach is illustrated in section 5.

In many practical scenarios precise models of the domain may be unavailable. For example, we do not have a precise model of the response of the human body to various disease causing agents. Medical diagnosis is therefore typically not model-based, or at least not entirely. A common practice in such scenarios is to rely on *heuristic* knowledge about the domain. Human experts are asked to summarize their knowledge (gained through experience) in the form of qualitative *principles*. These principles can then be codified in a *knowledge-based system*. Thus the system models the diagnostic reasoning of human experts. The efficacy of this approach obviously depends on the faithfulness of the encoding as well as the quality of the experts' domain experience. Notice that these principles are as perceived by the expert and the actual system's behavior (in this case the automobile's electrical system) is not explicitly modeled. Hence this approach is of considerable practical value in domains where a precise model is unavailable. *Expert systems* are popular examples of such systems and are typically built through a careful, tedious, and often expensive process of knowledge engineering (Durkin, 1994; Puppe, 1993; Stefik, 1995). Knowledge engineering refers to the task of eliciting and codifying the knowledge of the domain expert in a form that can be used by the system. Some difficulties associated with knowledge engineering are: experts are often unable to articulate their reasoning process sufficiently precisely to be encoded in a form usable by a machine; different experts often have different ways of approaching the same problem; lack of precision, combined with the fact that diagnostic rules of thumb are elicited from experts at different points in time (and perhaps in different contexts) can often lead to internal inconsistencies in an evolving knowledge base. Consistency maintenance in large scale evolving knowledge bases remains an open research problem. A simple example of an expert system for diagnosis is presented in section 4.

One approach which attempts to circumvent the difficult task of extracting and codifying the domain knowledge of the expert relies on building a large repository of sample diagnoses or *cases* (Kolodner, 1993). When presented with a diagnostic problem the system attempts to solve it by identifying one or more scenarios with known diagnoses from its repository of cases. Unlike model-based systems and expert systems, case-based systems neither model the domain knowledge nor the diagnostic reasoning of the domain expert. Instead, the knowledge is implicitly represented in the repository of cases. Clearly, the performance of case-based systems critically depends on the adequacy as well as organization of the collection of cases and the procedures used for searching the collection for cases that most closely *match* a given diagnostic scenario. A simple case-based diagnosis system is presented in section 6.

The difficulties associated with knowledge engineering have, in recent years, stimulated a great deal of research in *learning* systems. A learning system is essentially one that is capable of improving its performance at a given task (or a set of tasks) through experience (Honavar, 1994; Langley, 1995; Michalski, 1983; Mitchell, 1997; Simon, 1983; Uhr, 1973). A wide variety of learning systems are in common use: symbolic artificial intelligence systems (Buchanan & Wilkins, 1993; Langley, 1995; Mitchell, 1997; Shavlik & Dietterich, 1990), artificial neural networks or connectionist networks (Gallant, 1993; Hassoun, 1995; Kung, 1993), statistical pattern classification systems (Duda & Hart, 1973; Fukunaga, 1990; Ripley, 1996), syntactic pattern classification systems (Fu, 1982; Gonzalez & Thomason, 1978; Miclet, 1986), evolutionary systems (Goldberg, 1989; Holland, 1992; Koza, 1992; Michalewicz, 1992; Mitchell, 1996), etc. A detailed discussion of the different approaches to machine learning is beyond the scope of this paper. The interested reader is referred to (Honavar, 1994; Hutchinson, 1994; Langley, 1995; Mitchell, 1997) for such a discussion. Of particular interest for diagnosis applications are *inductive* learning systems. Unlike systems crafted through knowledge engineering, inductive learning systems have the ability to *extract* the domain knowledge from examples of problem-solving behavior. For example, if we have a case history of user reported problems and the diagnosis proposed by an expert for each of those cases, we could use an inductive system to extract the principles that approximately model the relationships between the problems and the corresponding diagnoses. Inductive approaches are particularly suited for domains in which principles are hard to formalize (weather prediction, medical diagnosis etc.) or where experts are few (space exploration etc.). Some widely used inductive learning systems include those based on decision trees (Quinlan, 1993) (see section 8), neural networks (Gallant, 1993; Hassoun, 1995) (see section 9), and statistical pattern classification (Duda & Hart, 1973; Fukunaga, 1990; Ripley, 1996) (see section 10).

4 EXPERT SYSTEMS

Expert systems (Durkin, 1994; Puppe, 1993; Stefik, 1995) are programs that model the expertise (knowledge) and reasoning capabilities of qualified specialists within fairly narrow domains (e.g., diagnosis of heart diseases, credit evaluation, etc.). Such systems are typically composed of three essential modules: a *knowledge base* that captures the expertise of the specialist, an *inference engine* that mimics the specialist's reasoning process, and a *working memory* that is used as a scratch pad. In the course of a problem-solving session, the working memory holds the facts provided by the user (e.g., symptoms in a diagnosis task) and intermediate conclusions derived by the inference procedure (Durkin, 1994; Stefik, 1995). A schematic of an expert system is shown in figure 1.

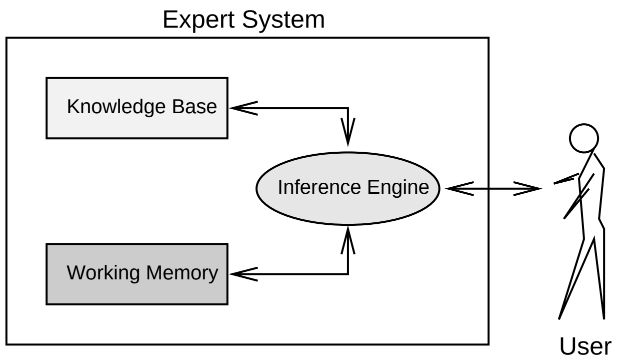

Figure 1: Schematic of an expert system. The knowledge base contains the domain knowledge encoded in some appropriate form (e.g., rules). In a consultation session with the user the inference engine generates inferences using the facts provided by the user and the rules in the knowledge base. The working memory holds user-provided facts and serves as a store for inferences or conclusions drawn based on the facts.

Critical to the process of expert system design is the task of knowledge engineering. A knowledge engineer interviews experts to obtain qualitative knowledge from them and codes it in the knowledge base using some representation scheme. Typical examples of knowledge representation schemes used in expert systems are: rules, semantic networks, frames, logic etc. For example, rules assert the known relations between *premises* and *conclusions*. In our automobile diagnosis example, we could formulate the domain knowledge about non-functioning Headlights in terms of the following rules:

Rule 1: IF Headlights don't work
THEN faulty Bulbs or/and Battery

Rule 2: IF faulty Bulbs or/and Battery
AND Engine does not start
THEN faulty Battery

Rule 3: IF faulty Bulbs or/and Battery
AND Engine starts
THEN faulty Bulbs

During a consultation with an expert system the user enters facts regarding a current problem into the working memory. The system matches these facts with the knowledge contained in the knowledge base to infer new facts. These new facts are then entered into the working memory and the process continues till the system completes the task presented to it or runs into a dead end. For instance in our automobile example, if the user reports a problem with his headlights while the engine starts just fine, then these two observations become a part of the initial working memory of the expert system for that consultation session.

The inference engine mimics the expert's reasoning process. It works from the facts in the working memory and uses the domain knowledge contained in the knowledge base to derive (or infer) new facts. It achieves this by searching through the knowledge base to find rules whose premises match the facts contained in the working memory. If such a match is found, it simply adds the conclusion of the matching rule to the working memory. This process continues until the inference mechanism is unable to match any rules with the facts in the working memory. In practice, inference process can be further complicated for a number of reasons. For instance, when multiple rules match the data in the working memory, mechanisms have to be provided for conflict resolution. Typical conflict resolution mechanisms are based on specificity of the rules (more specific rules, i.e., those that require more premises to hold, over more general ones); recency of the facts matched by the rules (as determined by the time of their entry into the working memory) etc.

Ideally, the inference procedure used by the system must be both *sound* and *complete*. Soundness and completeness are desirable formal properties of reasoning systems. Loosely speaking, soundness ensures that the conclusions drawn by the system are true in all the scenarios in which the facts on which they are based are true. Completeness ensures that the inference procedure is able to derive all true conclusions that follow from a set of facts and rules (Ginsberg, 1993; Russell & Norvig, 1995). Sometimes soundness and/or completeness of inference has to be sacrificed for efficiency.

Expert systems can be distinguished based on the inference strategy (forward-chaining from facts to conclusions, backward-chaining from hypotheses to premises, etc.), problem-solving strategy (bottom-up as opposed to top-down), knowledge representation (rule-based, frame-based, case-based etc.), inference techniques (deductive, non-monotonic, probabilistic, fuzzy, etc.), and their problem-solving paradigm (analysis: as in the case of diagnosis systems, versus synthesis: as in the case of design systems), etc. Their detailed treatment is beyond the scope of this paper. The interested reader is referred to (Durkin, 1994; Puppe, 1993; Stefik, 1995) for details.

As an illustration, let us assume that figure 2 constitutes the complete knowledge base for our automobile example (we assume that these rules were constructed somehow through a process of knowledge engineering).

Now, suppose a user reports a problem wherein the Headlights don't work. The expert system adds this fact to its working memory and then looks for a rule in the knowledge base that matches this data. It finds rule 1 and adds the conclusion, namely faulty Bulbs or/and Battery to the working memory. The inference engine cycles through the knowledge base again looking for a match for this newly added data in the working memory. It finds rule 2, but rule 2 requires another premise also to be satisfied, namely Engine does not start. Hence, the expert system queries the user about the status of the engine. Now suppose the user responds with the fact that the Engine does not start, the inference engine uses this new information to match rule 2, thereby adding the conclusion faulty Battery to the working memory. Further attempts to match this newly added information with rules in the knowledge base fail. Hence the inference procedure stops with the conclusion that the user is experiencing problems because of a faulty Battery. If the user had responded with Engine starts instead, rule 3 would have matched, leading to a conclusion of faulty Bulbs. This is essentially how an expert system functions. Notice that in the preceding discussion we have side-stepped issues like conflict resolution.

Our knowledge base does not support diagnoses involving multiple faulty components. Suppose the above diagnostic scenario had access to further information regarding the wipers. For instance, suppose we know in addition that the Wipers work. This changes the diagnosis completely, since working wipers vouch for a healthy battery. Thus in this case failure of headlights and engine must be due to faulty Ignition AND faulty Bulbs rather than a faulty Battery. However, irrespective of the sequence in which the symptoms become available, this diagnosis cannot be derived from our knowledge base. This is a common problem in expert system design and care must be taken to make sure that the rules in the knowledge base adequately cover the domain of interest.

Expert system design is a challenging task for a number of reasons. First, the experts

```
1.  IF      Headlights don't work
    THEN  faulty Bulbs or/and Battery
2.  IF      faulty Bulbs or/and Battery
    AND    Engine does not start
    THEN  faulty Battery
3.  IF      faulty Bulbs or/and Battery
    AND    Engine starts
    THEN  faulty Bulbs
4.  IF      Engine does not start
    THEN  faulty Battery or/and Ignition
5.  IF      faulty Battery or/and Ignition
    AND    Headlights work
    THEN  faulty Ignition
6.  IF      Wipers don't work
    AND    Engine does not start
    THEN  faulty Battery
7.  IF      Wipers don't work
    AND    Engine starts
    THEN  faulty Wiper Motor
8.  IF      Wipers don't work
    AND    Headlights work
```

Figure 2: The knowledge base of an expert system for the automobile diagnosis problem. Domain knowledge is coded in the form of IF-THEN rules. Inferences are drawn by matching the IF part of the rules with the known facts in the working memory and adding the conclusions back to the working memory. These conclusions represent partial inferences. This procedure is repeated until no further rules match the inferences in the working memory.

providing the domain knowledge must be able to articulate their expertise. In our automobile diagnosis example, the expert must be able to put down the relationships between the different automobile components and their dependencies. Second, the knowledge engineer working with the experts should be able to understand the expert's knowledge well enough to choose a good representation scheme to be used in the knowledge base and must determine an appropriate inference mechanism to work with the representation chosen. In our example, *rules* were chosen as the representation scheme. Third, the design of the knowledge base and the inference mechanism, the conflict resolution strategies used, and the order in which facts enter working memory can interact in fairly complex ways making the task of testing and debugging an expert system extremely difficult in practice.

5 REASON MAINTENANCE SYSTEMS

In contrast to expert systems where the reasoning process of the domain expert is captured, model-based systems attempt to model the fundamental operating principles of the domain. Although a complete model of the domain is unavailable or unwieldy to construct in many practical diagnostic scenarios (e.g., medical diagnosis), this approach is nevertheless useful in domains where the underlying physical principles are largely known (e.g., in artificial systems). For instance, automobiles are designed and manufactured based on sound engineering principles which in turn are grounded in physics. This can be used to build a detailed model of the automobile.

A model-based system for diagnosing automobile component failures can be built by formulating rules to capture the causal, functional knowledge of this domain, i.e., rules that relate the components of the automobile to the observable functions that they manifest. For example, domain knowledge in our automobile diagnosis task might be represented as causal rules of the form: if OK(Battery) AND OK(Ignition) THEN Start(Engine). This rules captures the working principle of a starting engine and relates it to the battery and ignition components of the automobile. Contrast this rule with the diagnostic rules of the expert system shown in figure 2. Since these rules model the functioning of the automobile by *assuming* properly working components, if some aspect of the automobile does not function properly, for instance, if the engine does not start, then the probable *cause(s)* can be easily identified from the rules. In this case it is a faulty battery or a faulty ignition. Reason maintenance systems (RMS) [also known as truth maintenance systems (TMS)] can be used to implement model-based diagnosis systems.

RMS systems typically contain a database of rules that represent the known facts and the working principles of the domain. Some facts may be known to be true while others might be assumed to be true. Based on these facts and assumptions, the rules may be used to draw inferences. However, if certain assumptions which were made earlier turn out to be false (in light of further evidence), the database must be updated accordingly. Not only will the truth of the assumptions change, but inferences that were drawn based on those assumptions must also be retracted or revised. RMS systems retract assumptions and revise inferences based on the current knowledge of the state of the world, thereby maintaining the soundness of the inferences in the database. RMS systems come in many different varieties (Forbus & de Kleer, 1993; Ginsberg, 1993), distinguished based on how they perform the task of keeping the database contradiction-free, and the kinds of inferences they support. In what follows, we will describe one approach to model-based diagnosis using an assumption-based truth maintenance system (ATMS).

Consider the following example. Suppose Mr. A has been invited to Ms. B's birthday party. Not a person to go empty handed, A looks around for an appropriate, yet affordable, gift. He decides to get her a bouquet of flowers. Even in this choice he makes an implicit assumption that B would like flowers and hence would be able to appreciate his gift. This sounds quite reasonable, since B is a woman, and women normally like flowers. If however, he suddenly runs into C, a close friend of B, and comes to know that B is allergic to flowers, he must to retract his assumption regarding B liking flowers, thereby also retracting the conclusion he had reached, namely to buy a bouquet of flowers as a gift. This done, he makes up his mind to get her a nice pair of earrings. This is an instance of assumption-based reasoning that can be performed by an ATMS. We will describe one such ATMS formulation based on Ginsberg (Ginsberg, 1993), and use it to illustrate the process of model-based diagnosis. But first, some essential details.

Suppose we have a database $\mathcal{D}$ of rules describing our domain. Further, suppose the database is partitioned into a set $\mathcal{C}$ of *common knowledge* (facts and rules) and a set $\mathcal{A}$ of *assumptions*, as shown in figure 3. Therefore $\mathcal{D} = \mathcal{C} \cup \mathcal{A}$. By an *explanation* for a sentence p we mean a *minimal* subset $\mathcal{E}$ of $\mathcal{A}$ such that $\mathcal{E}$ along with $\mathcal{C}$ is sufficient to entail the conclusion p. Mathematically, $\mathcal{E} \cup \mathcal{C} \models p$. (Note that $x \models y$, read x *entails* y, simply means that y is true in every scenario (or world model) in which x is true.) Loosely speaking, an explanation for a sentence p is simply a minimal set of assumptions that allow p to be derived using the facts and rules in $\mathcal{C}$. Suppose a new fact q becomes available. Since the database contains assumptions, some of which may contradict the new fact q, those assumptions (and inferences based on them) will have to be identified and retracted. Only then can the new fact q be safely added to the database. To do this, we first find all the explanations for $\neg q$ (the complement of q) in $\mathcal{D}$. Call these explanations $\mathcal{E}_1, \cdots, \mathcal{E}_k$. Next we find a *minimal* set $\mathcal{H}$ such that $\mathcal{H} \cap \mathcal{E}_i \neq \emptyset$, for every $\mathcal{E}_i$. Thus, $\mathcal{H}$ is the minimal set of assumptions that contradict the new fact q. Since these assumptions have to be retracted, we remove $\mathcal{H}$ from $\mathcal{A}$. Once these assumptions are retracted from the database, we can add the new fact q to $\mathcal{D}$ without introducing any contradictions.

Such a system can be easily adapted for diagnosis (Forbus & de Kleer, 1993). The database $\mathcal{D}$, in this case, is composed of a set $\mathcal{C}$ of facts describing the behavior of the system being diagnosed in terms of its components, and a set $\mathcal{A}$ of assumptions about the proper functioning of the components. For example, our automobile diagnosis problem, with sufficient causal knowledge (model knowledge) regarding the automobile, may be expressed as the database shown in figure 3.

Database entries 1, 2, 3 and 4 correspond to assumptions regarding the components, while entries 5, 6 and 7 capture the behavioral model of the domain. For example, entry 5 simply says that if the battery is OK and the ignition is also OK then the engine will start. Given this initial state, the system automatically infers Starts(Engine), Works(Headlights) and Works(Wipers) since they follow from the assumptions and the facts, and asserts them to be true, given the set of assumptions $\mathcal{A}$. In addition, each of these assertions is also labeled with the assumptions that support it. For example, Starts(Engine) would be labeled with (1, 4) since assumptions 1 and 4 are sufficient to derive it using the rules in $\mathcal{C}$.

$\mathcal{A}$: Assumptions and Inferences

1. OK(Battery)
2. OK(Bulbs)
3. OK(Wiper Motor)
4. OK(Ignition)

$\mathcal{C}$: Facts and Rules

5. OK(Battery) $\wedge$ OK(Ignition) $\rightarrow$ Starts(Engine)
6. OK(Battery) $\wedge$ OK(Bulbs) $\rightarrow$ Works(Headlights)
7. OK(Battery) $\wedge$ OK(Wiper Motor) $\rightarrow$ Works(Wipers)

Figure 3: A database of an ATMS system for the automobile diagnosis example. $\mathcal{C}$ contains sentences that are known to be true and rules that model the domain. $\mathcal{A}$ contains assumptions and inferences made based on the assumptions. The inferences are typically labeled with the assumptions that support their validity. Thus at a later point if some of the assumptions are retracted, the dependant inferences can also be identified and retracted.

Now suppose a particular user problem comes to light. Suppose it involves the non-functioning of the headlights and the engine, while the wipers are known to work. In our system these correspond to the sentences: $\neg$Works(Headlights), $\neg$Starts(Engine) and Works(Wipers). Since Works(Wipers) is already asserted in the system, it causes no problem. However, in order to add $\neg$Works(Headlights) and $\neg$Starts(Engine) to the database, we have to find the explanations for Works(Headlights) and Starts(Engine) and retract a minimal set of assumptions to enable the new sentences to be added without introducing any contradictions. In our example, the only explanation for Works(Headlights) is $\mathcal{E}_1 = \{$OK(Battery), OK(Bulbs)$\}$ and the only explanation for Starts(Engine) is $\mathcal{E}_2 = \{$OK(Battery), OK(Ignition)$\}$. We cannot remove the assumption OK(Battery) from the explanations since doing so would cause the assertion Works(Wipers) to become false. Hence we choose $\mathcal{H} = \{$OK(Ignition), OK(Bulbs)$\}$ as the minimal set of assumptions to retract in order to support the current input scenario. This corresponds to a diagnosis of faulty Ignition AND faulty Bulbs. As can be observed, this is an extremely accurate diagnosis.

Although model-based systems have the ability to produce accurate diagnosis, they have several drawbacks which make them inapplicable in many real-world contexts. Their need for an accurate model of the domain may not be possible to satisfy in many cases. In addition, the computation of all diagnoses for an observed fault is known to be computationally intractable which makes this procedure impractical for most real-world diagnosis applications. One alternative is to use *focusing mechanisms* for computing just a few of the probable diagnoses (Forbus & de Kleer, 1993). Another possibility is to use known polynomial-time algorithms for computing the first k diagnoses (for a given k) (Mozetic, 1992). Alternatively, one might use a hybrid approach which refines partial or incomplete model-based diagnostic knowledge into heuristic diagnosis rules through inductive learning (see sections 7-10 below). In any event, ATMS is just one of several model-based approaches to diagnosis.

6 CASE-BASED REASONING SYSTEMS

In case-based reasoning (CBR) systems (Schank & Abelson, 1977; Schank, 1982) knowledge is stored in the form of *cases*, where a case is defined as a contextualized piece of knowledge representing an experience that teaches a lesson fundamental to achieving the goals of the reasoner (Kolodner, 1993). Thus, a case can be thought of as a situation that was experienced in the past and resulted in some relevant action. The cases are stored in a *library*, indexed appropriately to facilitate efficient retrieval of the cases. Given a current experience or situation, the attributes of the input are used to index into the case library and retrieve the best matching case (or set of cases) according to some suitably defined *matching criterion*. One case is then chosen from among those retrieved to be the solution and this choice can itself be performed in a number of ways (the best match, most general match, etc). Since the new situation may not match the old one exactly, the solution offered by the selected case may be modified to fit the new situation, a process known as *adaptation*.

As an example, consider the following simplified formulation of our automobile diagnosis problem as a CBR task where each case is a diagnostic situation experienced earlier. For instance, case 1 in figure 4 indicates an earlier experience where a diagnosis of faulty Ignition was offered when the engine failed to start and no information was available regarding the status of the headlights and wipers.

Figure 4 shows the case library organized as a single-level structure, referred to as *flat memory organization* in CBR literature. Given this memory organization, the matching procedure serially examines the cases in the library and retrieves the best matching case (or cases). Since serial matching can be time consuming for large case libraries, other schemes based on indexing or hash-table based associative recall are often used in practice. Alternatively, parallel retrieval using appropriate hardware can also be used to facilitate efficient retrieval of cases. Case libraries can also be built by using clustering methods to group together *similar* cases (where similarity is defined according to some suitable, possibly domain dependent metric) resulting in a *hierarchical memory organization*. Figure 5 shows a possible hierarchical organization of the case library for the automobile diagnosis example. Such memory organizations can substantially speed up the retrieval of the best matching case(s).

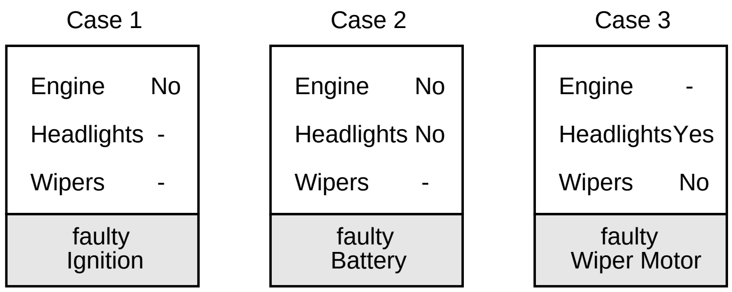

Figure 4: A simple case library for the automobile diagnosis problem. This library contains three cases corresponding to past diagnoses. In addition to using Yes and No to denote working and non-working components, the system also uses - to indicate the unknown status of components and symptoms.

Another important design decision involves the choice of *indices*. An index is a slot or field of a case that is used by the matching procedure in determining the degree of match between the input situation and a stored case. For example, the cases in our automobile example have three slots corresponding to the status of the engine, headlights, and the wipers. Indices have to be carefully chosen: The larger the number of indices used, the larger is the number of slots to be compared, and hence slower the retrieval. If the indices are fewer than necessary, precision of retrieval deteriorates. In our automobile example we have chosen all three slots — Engine, Headlights, and Wipers to be indices used for matching.

The matching (retrieval) procedure is probably the most critical component of a CBR system since it almost entirely determines what the system does with its stored knowledge. Expert knowledge about the domain is often used to construct *heuristic* matching rules that help the system return sensible matches for that domain. For our diagnosis problem, we might use a heuristic that unknown conditions ($-$) produce a weak match with yes or no values, identical values match strongly, and opposite values like yes and no do not match at all. This heuristic is demonstrated in table 1 where the values of 0, 1, and 2 stand for no match, weak, and strong matches respectively.

The entries in the table indicate the degree of match between a slot in the input situation and the corresponding slot value in a stored case. For example, suppose the input case has an entry yes corresponding to the slot Engine and we are trying to determine its match with a case in the library that has an entry no in the corresponding Engine slot, the degree of match between the two cases for the Engine slot is then a 0. The total degree of match between two cases can be defined as a simple *sum* of their degrees of match for the individual slots.

To clarify this further, suppose the new situation we observe is as shown in figure 6.

This input will match the stored cases to different degrees as shown in table 2.

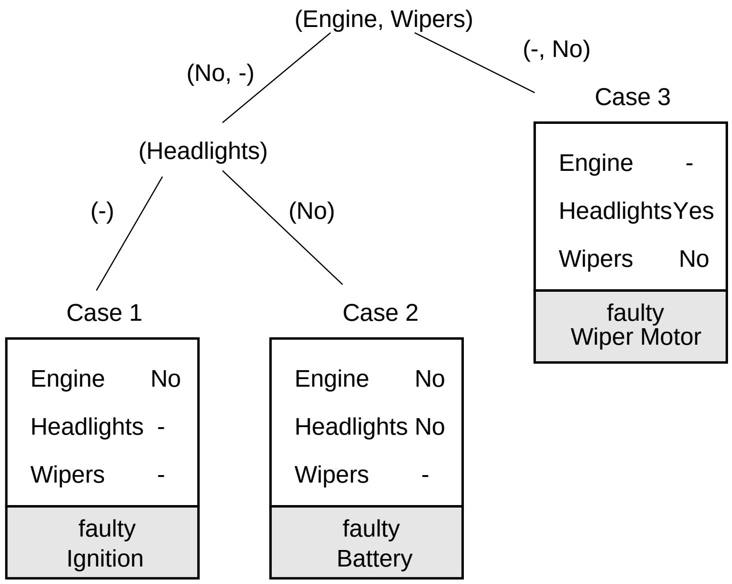

Figure 5: Hierarchical memory organization of the case library. A similarity-based clustering technique leads to a tree organization. The root of the tree performs a similarity clustering of the cases with like states of Engine and Wipers fields while the left subtree performs a sub-clustering based on the state of the Headlights. This results in a two-level hierarchy.

Based on the result of the matching process, case 2 is determined to be the best match with a degree of match 5. Consequently, a diagnosis of faulty Battery is proposed for the input situation of figure 6 since case 2 had a diagnosis of faulty Battery.

As it turns out this is an incorrect diagnosis since a faulty battery does not explain the correct functioning of the wipers. However, this is the best this CBR system can do as it has never experienced a situation corresponding to a diagnosis of faulty Bulbs. Allowing the system to adapt using feedback from the system designer or a domain expert gives it the ability to modify itself and hence produce better diagnoses over time. For example, suppose the above system obtains feedback from the designer to the effect that the proposed diagnosis was incorrect and that the right diagnosis in this situation should have been faulty Bulbs AND faulty Ignition, the system can adapt by incorporating this new knowledge in its case library though the addition of the case shown in figure 7. The next time around, this CBR system will be able to diagnose problems concerning faulty Bulbs. Such a system essentially learns by rote memorization. It should be noted that dynamic addition of cases to the library, might entail a restructuring of the library in CBR systems with hierarchically organized memory.

CBR systems differ significantly from knowledge-based expert systems and the model-based systems discussed in the previous sections. The diagnostic knowledge is neither stored as an explicit model of the domain nor in the form of qualitative, heuristic rules;

		Value of index in stored case		
		yes	no	–
Value of index in input case	yes	2	0	1
	no	0	2	1
	–	1	1	1

Table 1: A typical scoring function for matching cases. A value of 2 indicates a strong match between the corresponding values of the indices, while 1 indicates a weak match. Mismatched quantities have a score of 0. This matching function is applied to each index of the stored cases and the input situation.

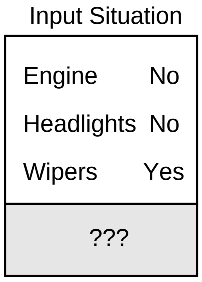

Figure 6: A new input situation for the automobile diagnosis problem. This diagnostic scenario corresponds to the case of non-functioning headlights and a non-starting engine while the wipers work. The CBR system is expected to produce a diagnosis for this situation.

rather, the knowledge is implicitly represented by the repository of cases. Problem-specific knowledge is used to design heuristics to guide the matching procedure (e.g., to generate the scoring function for matching cases, etc.). CBR systems have the ability (though limited) to adapt to changes within a domain since such changes simply result in a change in the stored cases. Note that with hierarchically organized case libraries, changes to cases might require reorganization of the entire library.

The drawbacks of CBR systems include the high computational cost associated with the matching procedure and the storage cost associated with the organization of the case library. One approach to avoiding such costs is to use inductive learning mechanisms (see section 7 below) to extract general rules that cover most of the cases in the repository. This can significantly reduce the size of the database of cases to a point where it needs to store only the known exceptions to the general rules. However, the applicability of such techniques is very dependent on the nature of the task domain and the representation of the cases. Alternatively, dedicated parallel hardware, e.g., neural network associative memories (Chen & Honavar, 1995; Chen & Honavar, 1996) may be used to speed up the

Case in Library	Degree of Match
1	4
2	5
3	1

Table 2: Degrees of match for the input case with each of the stored cases can be determined by using the scoring function shown in table 1. The degree of match is a sum of the scores obtained by matching each of the indices in the input and the corresponding stored case.

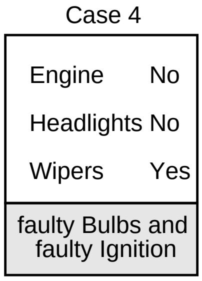

Figure 7: Adaptation in CBR systems. Here a new case is added to the case library. In this particular instance, a new diagnosis category is introduced. If the case library were organized in a hierarchical fashion, the addition of this new case might require a reorganization of the library.

similarity-based retrieval of cases.

7 INDUCTIVE LEARNING SYSTEMS

The systems discussed so far require the designer to incorporate the domain knowledge necessary for diagnosis either in the knowledge base or in the matching/inference process. Inductive learning systems offer a way to circumvent the difficult, expensive, and time consuming task of extracting such knowledge. When provided with examples (or pre-classified samples) of the domain, these systems have the ability to learn an approximate model of the task domain. In our automobile diagnosis task for example, an inductive system will be provided with examples of observed symptoms and their corresponding diagnoses possibly from a collection of past diagnoses performed by domain experts (like the cases in a case library). However, unlike CBR systems which simply store the examples, inductive systems use the examples to infer relationships between the symptoms and the diagnoses. Technically, the system *induces* an appropriate set of *classification rules* (Langley, 1995; Mitchell, 1997).

Formally, an example e_k is an ordered pair (I_k, C_k) where I_k is an instance, represented using some *instance language* like vectors of attribute values, strings over a fixed alphabet, etc. C_k is the classification of the instance I_k and is one of a set of possible classes $\mathbf{C}$ of the classification problem ($C_k \in \mathbf{C}$). Classes are normally described in a *concept language*. A *matching predicate* $M(I_k, C_j)$ provides a means for testing the membership of a given instance I_k in a class C_j (where $C_j \in \mathbf{C}$). Examples of concept languages include rules (of the sort used in expert systems), predicate logic formulae, automata (that accept or reject strings) etc. In the domain of diagnosis, we can think of instances as some encoding of observed symptoms and classes as representing the possible diagnoses. The problem of inductive learning is to learn a sufficiently accurate description of a concept or class $C_T \in \mathbf{C}$ from a sufficiently representative set of examples (and possibly counter-examples) of C_T. This essentially reduces to a search problem: namely, that of identifying a class description C_A that sufficiently closely approximates C_T. Various criteria (or *search biases*) might be used to guide this search through the concept space corresponding to $\mathbf{C}$. Inductive learning systems differ in terms of their choice of instance language, concept language, and search bias (among other things).

The set of examples provided by the user which enables the inductive system to determine a classification rule is called the *training set*. In our automobile diagnosis problem the company may have a set of such training examples based on past expert diagnosis of user reported problems. A sample training set is shown in table 3. (Note that this is only one of the many possible representations that could be chosen for encoding the training samples.)

Engine	Headlights	Wipers	Diagnosis
Yes	No	Yes	faulty Bulbs
No	No	No	faulty Battery
Yes	No	No	faulty Wiper Motor
No	Yes	No	faulty Ignition
Yes	Yes	No	faulty Wiper Motor
No	Yes	Yes	faulty Ignition

Table 3: Example diagnoses for the automobile diagnosis problem which constitutes the training set for inductive learning systems. The first three columns represent symptoms, and the last column indicates the diagnosis offered. Thus, each row of the table represents one diagnostic situation experienced earlier and successfully diagnosed. In these examples, Yes indicates *working* and No indicates *non-working* components.

For instance, the first entry simply means that there was a recorded case of an automobile electrical problem wherein the headlights failed to work. The engine had no problem starting and the wipers worked as well. In that particular case, the problem was diagnosed as faulty Bulbs. Similarly, the second entry says that in an observed case when the engine, headlights, and the wipers failed to work, the problem was diagnosed as being due to a faulty Battery, and so on. The entries of table 3 serve as training examples, and can be used by an inductive learning system to determine an appropriate classification rule or a set of rules that can assign a given instance of symptoms to one of the diagnosis classes.

In practice, the success of an inductive learning system depends on several factors. Perhaps the most important among these is the choice of instance and concept representation languages. In the training set shown in table 3, the observed problems are diagnosed as being caused by a single faulty component. For instance, the third example suggests that a failure of wipers and headlights is the result of a faulty wiper motor. Notice that this diagnosis does not explain the non-working headlights. In this case the correct diagnosis should have been faulty Wiper Motor AND faulty Bulbs. However, the concept language chosen is only capable of representing single faulty components. Thus any inductive procedure that uses the training examples shown in the table 3 cannot be (and should not be) expected to produce diagnosis involving multiple faulty components. However, in some cases multiple faults can be identified. For example, suppose an inductive system has been trained to classify the training examples of table 3 correctly. Now, if the diagnostic scenario corresponding to entry 3 of the training set repeats, the system will attribute it to a faulty wiper motor. Once the wiper motor is fixed, the diagnostic scenario would change as the wipers presumably, start working. The new observation would thus correspond to working engine and wipers and non-working headlights. The system will diagnose this new situation as due to faulty bulbs, thereby identifying both the faults in the original scenario. Thus, diagnosis involving multiple faulty components can be addressed, in some cases, by sequentially isolating single faults. However, this approach may not work in all cases.

One way to avoid this problem is to use a different concept language. For example, our diagnosis classes could be represented using a 4-tuple: (BulbStatus, BatteryStatus, WiperMotorStatus, IgnitionStatus), resulting in the training set shown in table 4.

However, in order to keep the size of the training set manageable and the calculations from becoming cumbersome, we will use the training set (and hence the concept language) of table 3 to illustrate the inductive learning approaches. (The concept language used in table 4 yields $2^4 = 16$ diagnosis classes. In order to learn reliably in this setting, we need a fairly large training set or prior knowledge.) One consequence of this choice is the fact that the learning systems have no way to represent (and hence learn) diagnoses that correspond to multiple faulty components.

Assuming an adequate choice of instance and concept languages, the success of an inductive learning algorithm also depends on the examples provided. If the examples are not *representative* of the domain of interest, the classification rules learned may not

Engine	Headlights	Wipers	Diagnosis
Yes	No	Yes	(faulty Bulbs, OK Battery, OK Wiper Motor, OK Ignition)
No	No	No	(OK Bulbs, faulty Battery, OK Wiper Motor, OK Ignition)
Yes	No	No	(faulty Bulbs, OK Battery, faulty Wiper Motor, OK Ignition)
No	Yes	No	(OK Bulbs, OK Battery, faulty Wiper Motor, faulty Ignition)
Yes	Yes	No	(OK Bulbs, OK Battery, faulty Wiper Motor, OK Ignition)
No	Yes	Yes	(OK Bulbs, OK Battery, OK Wiper Motor, faulty Ignition)

Table 4: The concept language used here (4-tuples) is capable of representing diagnoses involving multiple faulty components. For example, the third entry corresponds to a diagnostic scenario wherein the wiper motor and the headlight bulbs are both faulty.

characterize the domain appropriately. For example, if our training set of table 3 did not have any examples of diagnoses involving faulty bulbs, then the classification rules learned by the system cannot be expected to correctly diagnose the problems attributable to a faulty bulb.

Once the inductive system has been trained, i.e., it has induced the appropriate classification rules, it can be used to classify instances, including ones that were not seen while training. The ability of the system to classify previously unseen examples is referred to as *generalization*. For example, having trained an inductive system using examples from the above table, we might test the system with the unseen example shown in table 5 and expect a reasonable diagnosis like faulty Ignition and/or faulty Bulbs.

Engine	Headlights	Wipers
No	No	Yes

Table 5: Generalization is the ability of systems to correctly classify instances not seen while training. For example, a system trained with the entries of table 4 when presented with this unseen example, might be expected to produce a reasonable diagnoses like - faulty Ignition

Many factors affect the generalization ability of an inductive system including: the number and statistical distribution of the training examples, the search bias employed by the system, the concept language and instance language used, etc. This is a topic of considerable ongoing theoretical as well as experimental research. A detailed discussion of this and other related questions is beyond the scope of this paper. The interested reader is referred to (Buchanan & Wilkins, 1993; Gallant, 1993; Hassoun, 1995; Kung, 1993; Langley, 1995; Mitchell, 1997; Natarajan, 1991; Quinlan, 1993; Ripley, 1996; Shavlik & Dietterich, 1990) for discussions of theoretical and practical aspects of inductive learning systems. In what follows, we will restrict ourselves to a few simple examples of inductive learning systems that are well-suited for diagnosis applications. These are: *decision trees*, some simple classes of *artificial neural networks* (or connectionist networks), and some elementary *statistical approaches*.

8 DECISION TREES

The ID3 algorithm (Quinlan, 1993) provides a way to learn classification rules represented in the form of a *decision tree*. A decision tree consists of *leaf nodes* that are class names and *internal nodes* that represent tests on symptoms (also termed attributes), with a branch for each possible outcome of the test. For instance, our simple diagnosis problem might be solved using the decision tree shown in figure 8, where the tree first tests the headlights and then depending on the result of the test either tests the wipers next or the engine.

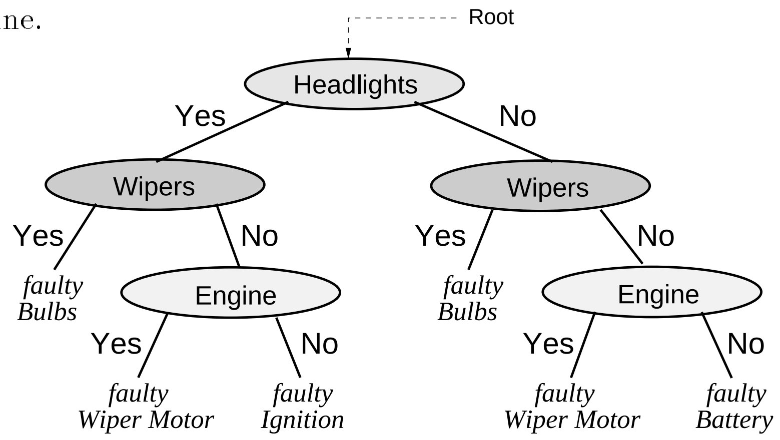

Figure 8: An example decision tree for the automobile diagnosis problem. A Yes denotes a working component and No a non-working one. Each node (shown as an ellipse) represents a test on the attribute serving as its label. Leaves represent faulty components and hence the diagnosis. A classification of an input situation is performed by starting at the root of the decision tree and testing the attributes in order until a leaf node is reached. The leaf node reached signifies the diagnosis (classification) of the input situation.

Once the decision tree is constructed, it can be used to classify input examples. To do so, we start at the root of the tree, evaluate the test specified there on the example to be classified and then take the branch corresponding to the appropriate outcome. We proceed with this evaluate-branch process until a leaf-node is encountered, at which point, the example is assigned to the class represented by that leaf-node. Consider the instance shown in table 5. To classify this instance (diagnose the cause of this problem), we first perform the test represented by the root of the tree, i.e., check whether **Headlights** work or not. Since in this instance, the headlights don't work, we follow the right branch of the tree and test if the **Wipers** work. Since in this instance they do, we branch left and reach a leaf node. Since this leaf node corresponds to a diagnosis of **faulty Bulbs**, we diagnose the problem as due to **faulty Bulbs**. Note that this decision tree requires two or more tests to classify examples. Since this decision tree was constructed somewhat arbitrarily, it may require more tests than necessary. As we will see later, ID3 constructs a more compact decision tree.

As long as the training set does not have *contradictory* examples (e.g., two training samples with exactly the same values for each of their attributes but assigned to two different classes), it is always possible to construct a decision tree that classifies all the training examples correctly. Usually there are many such decision trees that are consistent with a given set of examples and the object of induction is to somehow determine a decision tree that not only classifies the training examples correctly, but also has a high generalization accuracy. Quinlan's ID3 algorithm is based on an idea which is generally attributed to the English philosopher William of Ockham. This idea, known as Occam's razor can be roughly paraphrased as follows: given a set of observations that need to be explained, one must choose the *simplest* theory or model that adequately explains the observations. In decision tree construction, it simply means that given a set of examples the procedure should construct, in some reasonable sense of the term, the simplest decision tree that correctly classifies the training set. For example, we might choose a relatively small decision tree as opposed to a larger one so long as both trees correctly classify the training set. In addition to their greater likelihood of generalization, small decision trees require fewer attribute tests on average, leading to reduced computational cost of classification. The general idea behind the ID3 (Quinlan, 1993) algorithm is to recursively select attributes that yields the maximum possible amount of information (in the technical sense of the term used by Shannon in his development of information theory) on the average, for unambiguously classifying the training examples.

We will explain the main ideas by focusing on a two category classification problem. However, this approach can be extended naturally to multiple classes as will be demonstrated using the automobile diagnosis example. ID3 computes the expected information (number of yes/no questions that need to be asked) required to classify the examples in the training set, which can be shown to be (refer (Shannon, 1948) for details):

$$I(n_1, n_2) = -\frac{n_1}{n_1+n_2} \times log_2\left(\frac{n_1}{n_1+n_2}\right) - \frac{n_2}{n_1+n_2} \times log_2\left(\frac{n_2}{n_1+n_2}\right) \tag{1}$$

where n_1 and n_2 are the number of examples in the training set that belong to class 1 and class 2 respectively.

Now, if attribute A of the input takes on values $\{A_1, A_2, .., A_v\}$ and is to be used as the root of the decision tree, then the expected information required for the tree with A as the root, i.e., the expected information needed *after* choosing A as the root, is defined as:

$$E(A) = \sum_{i=1}^{v} \frac{n_1^i + n_2^i}{n_1 + n_2} \times I(n_1^i, n_2^i) \tag{2}$$

where n_1^i and n_2^i are the number of examples of the training set that belong to class 1 and class 2, and that have a value of A_i for their attribute A. The information gained by testing the value of attribute A is therefore:

$$gain(A) = I(n_1, n_2) - E(A) \tag{3}$$

At each step in the tree construction process ID3 computes the information gain for all *untested* attributes and chooses the attribute with the maximum information gain as the test for that particular node of the tree. Equivalently, we could choose the attribute with the minimum E.

To clarify this procedure, consider our automobile diagnosis problem. Here we are provided with training examples from table 3. This problem has three attributes: the functional status of the Engine, Headlights, and the Wipers. In addition, this problem has four output classes: faulty Bulbs, faulty Battery, faulty Wiper Motor, and faulty Ignition, which we label 1, 2, 3, and 4 respectively. We can easily determine the number of examples belonging to each class, for example, there is one example belonging to class 1 (Bulbs), one belonging to class 2 (Battery), while two examples each for classes 3 and 4 (Wiper Motor and Ignition). Therefore the expected information needed to classify the training examples using Eq. (1) is:

$$I(n_1, n_2, n_3, n_4) = -\frac{1}{6} \times log_2\left(\frac{1}{6}\right) - \frac{1}{6} \times log_2\left(\frac{1}{6}\right) - \frac{2}{6} \times log_2\left(\frac{2}{6}\right) - \frac{2}{6} \times log_2\left(\frac{2}{6}\right)$$

Which gives us:

$$I(1,1,2,2) = 1.9183$$

Now let us calculate the expected information of the three attributes — Engine, Headlights, and Wipers using Eq. (2) to enable us to choose the root of the tree.

$$E(\mathsf{Engine}) = \underbrace{\frac{0+1+0+2}{6}\left\{-\frac{1}{3}log_2\left(\frac{1}{3}\right) - \frac{2}{3}log_2\left(\frac{2}{3}\right)\right\}}_{\text{Engine does not start}} + \underbrace{\frac{1+0+2+0}{6}\left\{-\frac{1}{3}log_2\left(\frac{1}{3}\right) - \frac{2}{3}log_2\left(\frac{2}{3}\right)\right\}}_{\text{Engine starts}}$$

Using Eq. (3) we can calculate the gain as:

$$gain(\mathsf{Engine}) = I(1,1,2,2) - E(\mathsf{Engine}) = 1.9183 - 0.9183 = 1.0$$

Similarly we can determine

$$gain(\mathsf{Headlights}) = 1.9183 - 1.2516 = 0.6667$$

and

$$gain(\mathsf{Wipers}) = 1.9183 - 1.3333 = 0.5850$$

Since the rule of thumb is to maximize the information gain, we choose Engine as the test for the root node. We then repeat this process by partitioning the training examples into two disjoint sets corresponding to the left and right subtrees of the decision tree constructed so far, i.e., one set of examples with a starting engine and the other with a non-starting one. Now the procedure is repeated for the two branches each with their own set of examples. This leads us to the decision tree shown in figure 9, which can be seen to be more compact than the tree in figure 8. However, note that this tree produces a diagnosis of faulty Battery on the input situation of table 5, unlike the more appropriate diagnosis of faulty Bulbs of the earlier decision tree. Since working wipers vouch for a working battery, this diagnosis is incorrect. However, such generalization errors are not uncommon when the training set is not sufficiently representative of the domain of interest.

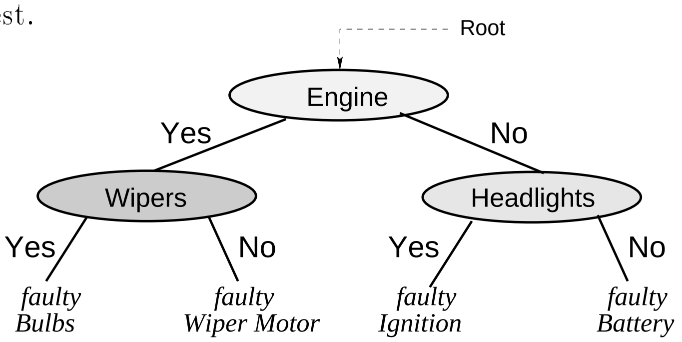

Figure 9: Decision tree determined by ID3. The information-theoretic approach of ID3 results in a more compact decision tree. This tree only requires two attribute tests in all cases while the one before required either two or three.

An obvious advantage of the decision tree approach to inductive learning is that it is extremely straightforward to transform the resulting tree into a set of equivalent IF-THEN rules of the sort used in expert systems. This makes it possible to use ID3-like algorithms to circumvent the need for difficult, tedious, and expensive knowledge engineering in the design of expert systems. For example, the right branches of the tree give us the following rule:

IF Engine does not start AND Headlights don't work
THEN faulty Battery

The decision-tree approach is fairly robust and can tolerate noisy and incomplete data within reasonable limits. Compared to other inference procedures (deduction and abduction systems), decision trees are much faster to execute. Some drawbacks associated with the use of decision tree construction algorithms such as ID3 and some possible remedies are documented in (Quinlan, 1993). Decision trees have been extended to handle continuous attribute ranges in the form of discretized, multi-valued intervals. Relatively straightforward extensions to decision-tree construction algorithms which seek to minimize the cost or risk associated with the task of diagnosis (e.g., when all tests are not equally expensive or risky as is often the case in medical diagnosis) can be formulated and are worth exploring.

9 ARTIFICIAL NEURAL NETWORKS OR CONNECTIONIST NETWORKS

Connectionist networks or *artificial neural networks* are massively parallel, shallowly serial, highly interconnected networks of relatively simple computing elements or *neurons* (Gallant, 1993; Hassoun, 1995; Honavar, 1994; Kung, 1993; Ripley, 1996). Much of the attraction of connectionist networks stems from their massive parallelism, ability to adapt through a modification of their computational structure, and their potential for reliable performance in the presence of noise or failure of components.

The input to an n-input neuron is typically represented by a pattern vector $\mathbf{X} \in \Re^n$ or in the case of binary patterns, by a binary vector $\mathbf{X} \in [0,1]^n$. Each neuron computes a relatively simple function of its inputs and transmits outputs to other neurons to which it is connected via its output links. A variety of neuron functions are used in practice, the most common ones being *linear*, *threshold*, *sigmoid*, and *radial-basis* functions. Each neuron has associated with it a set of modifiable parameters which are adapted through learning. Commonly used parameters are the so-called *weights*, that represent the strength of the synapses between the neurons. Additionally, each neuron is also assumed to be connected to a constant source (+1 or -1) and the weight on this link is called the *bias* or *threshold* of the neuron. The weights associated with an n-input neuron i is typically represented by an n-dimensional weight vector $\mathbf{W}_i \in \Re^n$. By popular convention, the first element of the weight vector is usually the bias. The input activation of a neuron i, denoted A_i, in response to a pattern $\mathbf{X}$ on its input links is given by the vector dot product: $\mathbf{W}_i \cdot \mathbf{X}$. The output of the neuron, O_i, a function of the input activation. For example, the threshold function produces $O_i = 1$ if $\mathbf{W}_i \cdot \mathbf{X} \geq 0$ and $O_i = -1$ otherwise.

A variety of connectionist networks have been studied in the literature. In contrast with feed-forward networks (where their connectivity graph does not contain any directed cycles), networks can contain feedback loops and such networks are referred to as recurrent. In addition, networks can be single-layered or multi-layered; sparsely connected or completely connected; strictly layered or arbitrarily connected; homogeneous or heterogeneous, etc. The computational capabilities (and hence pattern classification abilities) of a neural network depend on its architecture (connectivity), functions computed by the individual neurons, and the setting of parameters or weights used. Hence, designing a neural network for a particular pattern classification task involves determination of the network architecture (number of neurons, their connectivity, etc.), the types of neurons (e.g., linear, sigmoid, threshold, etc.), as well as the parameter or weight values. This is typically accomplished through a combination of design (using a-priori knowledge or guesswork) and learning (which may be used to modify the weights, network architecture, learning algorithm itself, etc.) (Gallant, 1993; Honavar & Uhr, 1993; Honavar, 1994; Parekh *et al.*, 1997).

Much of the research on neural network learning/training has focused on algorithms that modify the weights within an otherwise fixed network architecture. This essentially entails a search for a setting of weights that enable the network to classify all (or most of) the samples in the training set correctly. Since this is fundamentally an *optimization* problem, a variety of optimization methods (gradient-descent, simulated annealing, etc.) can be used to determine the weights. Most of the popular learning algorithms use some form of error-guided search (e.g., changing each modifiable parameter in the direction of the negative gradient of a suitably defined error measure with respect to the parameter of interest). For examples of such algorithms the reader is referred to (Gallant, 1993; Kung, 1993; Hassoun, 1995; Ripley, 1996).

We will illustrate the connectionist approach to our automobile diagnosis example by using a rather simple single-layer network. The following figure shows a neural network for our automobile diagnosis example. For the moment, let us assume that some procedure has generated the weights of the network (we will describe such a procedure later).

The network has three input neurons, one each for the status of Engine, Headlights, and Wipers, with the understanding that if a component works, the corresponding input neuron is provided a value of 1, and a -1 otherwise. The four output neurons of the network each correspond to a diagnosis category — faulty Bulbs, faulty Battery, faulty Wiper Motor, and faulty Ignition. The output neurons compete and the neuron with the highest activation is declared the winner. The winning neuron produces an output of 1, while the rest produce a -1. This scheme is called the *winner-take-all* strategy in connectionist literature, and it ensures that the network only diagnoses single faults (as stipulated by the concept language of table 3). If the output of the neuron labeled faulty Bulbs is a 1, it is interpreted to mean that the connectionist network has diagnosed that the problem is due to faulty bulbs. An output of -1 means that the particular component is not at fault. The numbers inside the output units represent the strength of the bias,

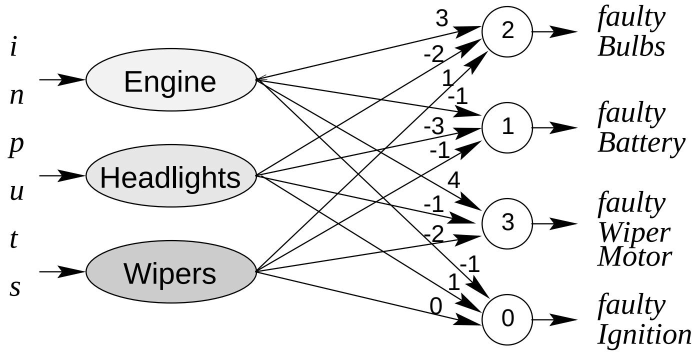

Figure 10: An example connectionist network for the automobile diagnosis problem constructed using perceptron learning. The network contains three input units and four output units that compete for domination. The unit with the maximum activation is declared the winner and it produces an output of 1. The other units produce an output of -1. The numbers inside the output units represent the additive bias. Input units receive a value of 1 if the corresponding component works and a -1 when it does not. A 1 at an output unit indicates that the corresponding component is faulty, while a -1 indicates a healthy component.

which in our example, is assumed to be connected to a constant source of +1.

Consider the performance of this neural network diagnosis system on a problem shown in table 5, which involves the observations that the engine and headlights don't work while the wipers do. The corresponding input to the network will be (-1, -1, 1) at the corresponding neurons for Engine, Headlights, and Wipers. Applying these inputs to the network in figure 10, we obtain the following:

$$A_{\mathsf{faultyBulbs}} = (2, 3, -2, 1).(1, -1, -1, 1) = 2 \times 1 + 3 \times -1 + -2 \times -1 + 1 \times 1 = 2$$

Note that in this calculation, the first component of the weight and input vectors corresponds to the bias of the unit. Similarly, we can determine the activations at the other output neurons.

$$\begin{aligned} A_{\mathsf{faultyBattery}} &= 4 \\ A_{\mathsf{faultyWiperMotor}} &= -2 \\ A_{\mathsf{faultyIgnition}} &= 0 \end{aligned}$$

Since the unit corresponding to faulty Battery has the strongest activation, it is declared the winner. Thus the network diagnoses that the battery is faulty. As we mentioned in the context of decision trees, this diagnosis is not really accurate since it does not explain the proper functioning of the wipers. This generalization error is an artifact of the limited number of training examples in our diagnosis example.

In this example we used a variant of Rosenblatt's perceptron learning rule (Gallant, 1993; Hassoun, 1995) to obtain the weights of the network. Starting with random initial weights, this procedure presents examples from the training set either randomly or sequentially, and updates the weights if the network's output differs from the one desired. The weight modification rule used in such circumstances, is given by the following equation:

$$\mathbf{W}_i(t+1) = \mathbf{W}_i(t) + \eta \times (D_i - O_i) \times \mathbf{X}_i \tag{4}$$

where $\mathbf{W}_i$ is the weight vector associated with neuron i, t indicates the time step. D_i and O_i represent the desired and currently obtained outputs of neuron i. $\mathbf{X}_i$ is the vector representing the input values of neuron i and η is the learning rate (typically chosen to be 1). Suppose $\mathbf{W}_{\mathsf{faultyBulbs}}(t) = (0, 1, 0, -1)$ and the first example of the training set is being presented. This example corresponds to $\mathbf{X}_{\mathsf{faultyBulbs}} = (1, 1, -1, 1)$. (Remember that the first component of the weight vector represents the bias and the corresponding component in the input vector is the source of $+1$.) Now, the output unit corresponding to faulty Bulbs will produce an activation: $A_{\mathsf{faultyBulbs}} = 0 \times 1 + 1 \times 1 + 0 \times -1 + (-1) \times 1 = 0$. Suppose that in this case some other output unit is the winner. Thus, $O_{\mathsf{faultyBulbs}} = -1$, contrary to the desired output $D_{\mathsf{faultyBulbs}} = 1$. The weight update rule of Eq. (4) is used to update the weight vector as follows:

$$\begin{aligned} \mathbf{W}_{\mathsf{faultyBulbs}}(t) &= (0, 1, 0, -1) + 1 \times (1 - (-1)) \times (1, 1, -1, 1) \\ &= (2, 3, -2, 1) \end{aligned}$$

Similarly, the weights of the neuron that won wrongly are also updated using the perceptron rule.

Figure 11 shows another network for the same diagnosis problem. However, in this case, the weights were determined by another learning algorithm known as Hebbian learning (Hebb, 1949; Hassoun, 1995; Kung, 1993). Hebbian learning is a simple, associative procedure given by Eq. (5).

$$\mathbf{W}_i = \sum_{p=1}^{P} \mathbf{X}_i^p \times D_i^p \tag{5}$$

where $\mathbf{W}_i$ is the weight vector of output unit i and P is the number of training patterns available to the system. $\mathbf{X}_i^p$ is the input vector of the ith unit when presented with the pth training example, and D_i^p is the desired output of neuron i for the pth training example. For example, the weights of the output neuron corresponding to faulty Bulbs can be determined as follows:

$$\begin{aligned} \mathbf{W}_{\mathsf{faultyBulbs}} &= (1, 1, -1, 1) \times 1 + (1, -1, -1, -1) \times (-1) + (1, 1, -1, -1) \times (-1) \\ &\quad + (1, -1, 1, -1) \times (-1) + (1, 1, 1, -1) \times (-1) + (1, -1, 1, 1) \times (-1) \\ &= (-4, 2, -2, 4) \end{aligned}$$

Similar calculations lead to the network in figure 11.

Notice that Hebbian learning is *one-shot*, i.e., the weights of the network can be determined by considering the training patterns just once, unlike perceptron learning

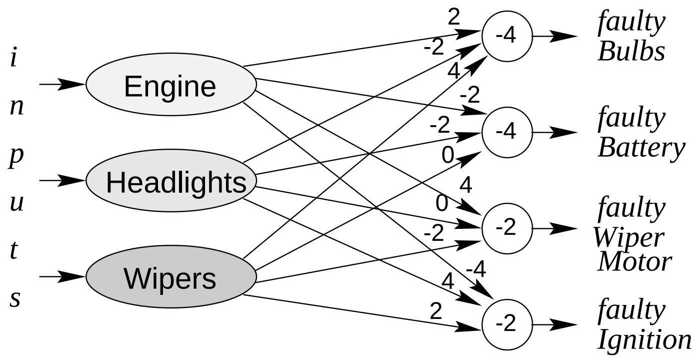

Figure 11: A connectionist network constructed using Hebbian learning.

which might require multiple passes to be made over the training set. It is interesting to observe the response of the Hebbian network to the input (-1, -1, 1). The activations produced are (0, 0, -8, 0). Since there is no clear winner, we could either randomly break the tie, or suggest that the user might have faulty Bulbs, or faulty Battery, or a faulty Ignition. Observe that this diagnosis is quite different from the ones produced by the diagnosis systems introduced earlier. This illustrates the fact that different learning systems, when presented with the same training data, can learn different classification rules. This is attributable to the different *inductive and representational biases* implicit in the design of the learning algorithms.

In summary, neural networks offer a powerful set of tools for inductive learning of diagnostic knowledge from examples (provided a representative set of examples is available or can be obtained). Their potential advantages include their ability to tolerate limited amounts of noise in their input, their ability to handle missing values, etc. For example, the connectionist networks could be provided with a representation of missing values with 0 denoting the unknowns. A main drawback of connectionist systems is that unlike expert systems and decision trees, their behavior can be hard to understand because of their reliance on numerical computation involving a large number of weights. However, techniques for extracting the knowledge learned by a neural network in rule-like form are becoming available (Gallant, 1993; Shavlik, 1994). Techniques for incorporating prior (possibly domain specific or heuristic) knowledge into connectionist networks are also being developed. Such knowledge can take a variety of forms. A number of researchers have investigated techniques of initializing a network with knowledge available in the form of propositional rules (Gallant, 1993; Shavlik, 1994). This opens up the possibility of a variety of hybrid schemes using combinations of expert systems and neural networks.

10 STATISTICAL PATTERN CLASSIFICATION SYSTEMS

Traditional AI systems have been limited in their applicability to many real world diagnosis problems because of their reliance on a *closed world* assumption. In such logic-based systems, a sentence s (which represents some fact concerning the world) is known to be either true or false at any given point in time, i.e., with probability 1 we know it to be true or false. However, in many practical scenarios, we may be hard-pressed to determine with absolute certainty the truth value of the sentence s. One approach to reasoning in such circumstances is to assume a truth value (either true or false) and proceed, as in the case of ATMS systems presented in section 5 with the possibility of retracting the assumption (and any inferences that relied on the assumption) in the light of subsequent evidence to the contrary. Probabilistic inference procedures offer a different approach to this problem by using probabilities to model uncertainty. For instance, we may know that sentence s has been true in 70% of the situations in the past. We can then characterize our belief in the truth value of s through the use of probabilities, i.e., $P(s) = 0.7$ (with probability 0.7 sentence s is true).

Statistical pattern classification systems (Duda & Hart, 1973; Fukunaga, 1990; Ripley, 1996) use probabilistic inference procedures that perform classification by making use of the statistical distribution of the attribute values of the patterns in each of the classes. Once this distribution has been estimated using the training examples, the attribute values of the instance to be classified can be used to compute the beliefs in the different classes, given the instance. The class with the strongest belief is then chosen as the classification of the instance. Technically, the chosen class is the *most likely* given the attribute values comprising the instance. The approach we will describe in this paper is based on an application of Bayes rule (Duda & Hart, 1973; Fukunaga, 1990; Ripley, 1996).

Consider a diagnosis scenario wherein inputs correspond to a set of observations (symptoms), represented by a vector of random variables $\mathcal{X} = (\mathcal{X}_1, \mathcal{X}_2, \cdots, \mathcal{X}_n)$. For example, in our automobile diagnosis task, $\mathcal{X}_1, \mathcal{X}_2$, and $\mathcal{X}_3$ might represent the observed status of the headlights, engine, and the wipers. Each of these symptoms take on specific values: $\mathcal{X}_1$ takes on the value x_1, $\mathcal{X}_2$ takes on the value x_2, etc., with certain probabilities. For example, one can examine the training set shown in table 3 and determine the fraction of instances in which which the engine starts, thereby obtaining the probability with which the variable corresponding to the engine, $\mathcal{X}_2$, takes on the value `starts`. Let $\mathbf{X} = (x_1, x_2, \cdots, x_n)$ represent a particular instance of the random variable $\mathcal{X}$ that is observed, which corresponds to a diagnostic scenario. That is, the observed values of the random variables $\mathcal{X}_1, \cdots \mathcal{X}_n$ are $x_1, \cdots x_n$ respectively. To avoid unnecessary clutter in the notation, we will use $P(\mathbf{X})$ to denote the probability that the random variable $\mathcal{X}$ takes the value $\mathbf{X}$. Similarly, we will use $P(x_j)$ to denote the probability that the random variable $\mathcal{X}_j$ takes the value x_j. The diagnosis system has the task of assigning the observed input $\mathbf{X}$ to one of m diagnosis categories, $C_1, C_2, \cdots, C_m$. For instance, in our automobile diagnosis problem, classes C_1, C_2, C_3, and C_4 might correspond to faulty Bulbs, faulty Battery, faulty Wiper Motor, and faulty Ignition respectively. Given a particular input instance $\mathbf{X}$, Bayes rule allows us to calculate the likelihood of occurrence of each of the categories by using Eq. (6).

$$\forall i \in (1, m), \quad P(C_i|\mathbf{X}) = \frac{P(\mathbf{X}|C_i) \times P(C_i)}{P(\mathbf{X})}, \tag{6}$$

where

$$P(\mathbf{X}) = \sum_{i=1}^{m} P(\mathbf{X}|C_i) \times P(C_i) \tag{7}$$

Once the probability of occurrence of the diagnosis classes C_i conditioned by the occurrence of $\mathbf{X}$ have been computed using Eq. (6), the classification of $\mathbf{X}$ can be performed by choosing the class C_k that produces the maximum likelihood, i.e., $\forall j \neq k, P(C_k|\mathbf{X}) \geq P(C_j|\mathbf{X})$.

Thus, Bayes rule can be used in diagnosis applications as long as we can calculate, either using domain knowledge or estimation procedures, the quantities on the right hand side of the expression in Eq. (6). Consider a scenario wherein the inputs are characterized by the random variables $\mathcal{X}_1, \mathcal{X}_2$, and $\mathcal{X}_3$ (as might be the case in our automobile diagnosis example). If the attributes are observed to take on values x_1, x_2, and x_3, Eq. (6) can be represented as follows:

$$P(C_i|x_1, x_2, x_3) = \frac{P(x_1, x_2, x_3|C_i) \times P(C_i)}{P(x_1, x_2, x_3)} \tag{8}$$

In order to use Eq. (8) for diagnosing the cause of symptoms x_1, x_2, and x_3, we must calculate the conditional and unconditional joint probabilities: $P(x_1, x_2, x_3|C_i)$ and $P(x_1, x_2, x_3)$ respectively. Estimating the joint probabilities is a difficult task in general, since it requires the evaluation to be made for all combinations of the attributes (observations), which is exponential in the number of attributes. Hence this procedure becomes intractable for problems involving a large number of attributes. By making simplifying assumptions regarding the independence of causal relationships, the calculation of the joint probabilities can be reduced significantly in number. For instance, *conditional independence* of the observations (symptoms) given that a particular diagnosis category is observed, is often assumed to hold. For example, $P(x_1|C_i, x_2) = P(x_1|C_i)$ implying that the symptoms x_1 and x_2 are independent given that class C_i is observed (or suspected) (Russell & Norvig, 1995). With such assumptions, Bayes rule for an m category problem involving 3 observations can be reduced to the expression in Eq. (9).

$$P(C_i|x_1, x_2, x_3) = \frac{P(x_1|C_i) \times P(x_2|C_i) \times P(x_3|C_i) \times P(C_i)}{\sum_{j=1}^{m} P(x_1|C_j) \times P(x_2|C_j) \times P(x_3|C_j) \times P(C_j)} \tag{9}$$

Let us illustrate this approach with our automobile diagnosis example. Consider the training examples of table 3. It is easy to determine the frequency distribution of the diagnosis classes. For instance, we can observe that $\frac{1}{6}$ of the user problems were attributed to faulty Bulbs, $\frac{1}{6}$ to a faulty Battery, $\frac{2}{6}$ to a faulty Wiper Motor, and finally, $\frac{2}{6}$ to faulty Ignition. These represent the prior probability of occurrence of each of the diagnosis classes.

$$P(\neg Bu) = 0.167,\ \ P(\neg Ba) = 0.167,\ \ P(\neg Wm) = 0.333,\ \ P(\neg I) = 0.333$$

where we denote Bulbs by Bu, Battery by Ba, Wiper Motor by Wm, and Ignition by I. Further, the prefix $\neg$ applied to a symbol indicates that the particular component is faulty.

As mentioned earlier, for Bayesian inference to work, we also need the class conditional probabilities, i.e., $P(x_j|C_i)$ of each of the observations x_j. In the automobile diagnosis example, we might have access to some amount of causal knowledge of the kind: if the battery is faulty the headlights won't work, i.e., $P(\neg H|\neg Ba) = 1$. Similarly, we can determine the other required class conditional probabilities from causal knowledge, where applicable. In many cases, there is no direct causal relationship between the fault and the symptom. For instance, faulty bulbs do not directly affect the functioning of the engine. In this case, how do we compute the conditional $P(\neg E|\neg Bu)$? In many real world scenarios, the absence of direct causal relationship between a fault and a symptom implies that they are independent. For example, in the automobile model we have assumed, the functioning of the bulbs is independent of the engine and vice versa. Thus we can compute the class-conditioned probability as: $P(\neg E/\neg Bu) = P(\neg E)$, by assuming independence of $\neg E$ and $\neg Bu$. Further, given the training examples, we can easily estimate the unconditional (or prior) probabilities of the observations, i.e., $P(\neg E)$ can be estimated from the examples. Hence $P(\neg E/\neg Bu) = P(\neg E) = \frac{3}{6} = 0.5$. This approach leads to the following class-conditioned probabilities:

$$\begin{array}{lll}
P(\neg H/\neg Bu) = 1 & P(\neg E/\neg Bu) = 0.5 & P(\neg W/\neg Bu) = 0.6667 \\
P(\neg H/\neg Ba) = 1 & P(\neg E/\neg Ba) = 1 & P(\neg W/\neg Ba) = 1 \\
P(\neg H/\neg Wm) = 0.5 & P(\neg E/\neg Wm) = 0.5 & P(\neg W/\neg Wm) = 1 \\
P(\neg H/\neg I) = 0.5 & P(\neg E/\neg I) = 1 & P(\neg W/\neg I) = 0.6667
\end{array}$$

Once these quantities are determined the system is ready for diagnosing problems. For example, consider the familiar user problem wherein the wipers work (W), but the headlights don't work ($\neg H$), and the engine does not start ($\neg E$). Using Bayes rule of Eq. (9), the system can diagnose the problem as follows:

$$P(\neg Bu/\neg H, \neg E, W) = \frac{P(\neg Bu) \times P(\neg H/\neg Bu) \times P(\neg E/\neg Bu) \times P(W/\neg Bu)}{\sum_{i=Bu,Ba,Wm,I} P(\neg i) \times P(\neg H/\neg i) \times P(\neg E/\neg i) \times P(W/\neg i)} \tag{10}$$

which becomes:

$$P(\neg Bu/\neg H,\neg E,W)=\frac{0.167\times 1\times 0.5\times 0.3333}{0.0278+0+0+0.0554}=0.3341 \tag{11}$$

Similarly one arrives at the following:

$$P(\neg Ba/\neg H,\neg E,W)=0 \tag{12}$$

$$P(\neg Wm/\neg H,\neg E,W)=0 \tag{13}$$

$$P(\neg I/\neg H,\neg E,W)=0.6659 \tag{14}$$

As Eq. (14) has the maximum $P(Diagnosis/\neg H,\neg E,W)$ value, the system diagnoses the input observations as due to a faulty Ignition. Notice that the input also has a non-zero likelihood of being an outcome of faulty bulbs.

A number of issues need to be dealt with to make use of probabilistic inference systems in practice. The primary problem with these systems is the determination of the conditional joint distributions which are exponential in the number of attributes. Algorithms for approximate computation of the conditional probabilities work quite well in practice (Dean *et al.*, 1995; Pearl, 1988). One such tool for efficient probabilistic reasoning is the *belief network* (Dean *et al.*, 1995; Pearl, 1988; Russell & Norvig, 1995). Belief networks (also called probabilistic networks or Bayesian networks) are structures that concisely represent the joint probability distributions of random variables. They contain nodes representing sets of random variables and directed links between nodes that capture the influences between them. Each node is equipped with a conditional probability table that quantifies the effect of parents of the node. A detailed exposition of belief networks is beyond the scope of this paper and the interested reader is referred to (Pearl, 1988; Russell & Norvig, 1995) for further details.

In summary, statistical approaches such as the probabilistic reasoning system outlined above rely on the availability of the necessary class-conditional probabilities (or a representative set of samples from which such probabilities can be estimated) and the prior-probability distributions. They constitute a powerful approach to diagnosis especially in noisy domains or in applications in which partial domain knowledge is available to establish an initial structure for Bayesian networks which can then be further refined using training data.

Other, not necessarily statistical, techniques for reasoning under uncertainty include Dempster-Schafer calculus, fuzzy logic, and related methods (Tanimoto, 1995). Dempster-Schafer theory is designed to deal with the distinction between uncertainty and ignorance. It is used to calculate the probability that the evidence (or observation) supports a conclusion, rather than the probability of the conclusion itself. It differs from probabilistic reasoning systems in this important regard. A belief function is used to measure the degree of belief in a particular state. This allows a system to hold beliefs regarding the state of the world, for example the state of an automobile component, without the beliefs necessarily being constrained to add up to 1 (unlike probabilistic systems). Dempster-Shafer theory provides a complete computational model for generating belief functions and updating them by combining new evidence (Dempster, 1968; Shafer, 1976).

Fuzzy logic captures the notion of *vagueness* by using fuzzy sets to denote how well an object or a situation satisfies a vague description. Rather than true/false values, fuzzy predicates take on truth values between 0 and 1. These systems thus model the fuzziness of class boundaries rather than the uncertainty in the probabilistic sense (Zadeh, 1965; Zimmermann, 1991). Like in classical logic which supports only two truth values, in probability theory, assertions about the world can be either true or false. However, each assertion is true or false with a certain probability. Fuzzy logic, on the other hand, allows one to associate a real-valued degree of truth with each assertion thereby supporting an infinite number of truth values.

11 SUMMARY AND DISCUSSION

The AI literature offers a wide variety of approaches to the design of intelligent diagnosis systems. In order to keep this paper concise and in some regards focused, we have chosen to limit the bulk of the discussion to only approaches that have been widely used in the design of intelligent diagnosis systems, or appear to hold considerable promise. We have illustrated each of the approaches by outlining the construction of a diagnosis system for identifying simple faults in an automobile electrical system.

Intelligent diagnosis, like any other task requiring intelligence, needs sufficient knowledge of the domain of interest. The choice of a particular design for an intelligent diagnosis system is therefore influenced greatly by the form, amount, or sources of knowledge that are available. If adequately complete and precise knowledge of the operating principles of the domain is available, it can be used to develop causal models which can support a model-based approach to diagnosis (Forbus & de Kleer, 1993). In the absence of such a causal model, if the knowledge necessary for diagnosis can be acquired from human experts, it would be possible, at least in principle, to build a knowledge-based system or an expert system for diagnosis (Durkin, 1994; Puppe, 1993; Stefik, 1995). In practice, such knowledge engineering can be expensive and time consuming.

In many practical scenarios, knowledge elicitation from experts may not be feasible. However, it might be possible to obtain a large sample of diagnoses performed by domain experts. These may be stored in a case library and used by case-based reasoning (CBR) systems (Kolodner, 1993). One can view such a system as performing a form of rote learning or memorization.

In recent years, much AI research has focused on automating the knowledge acquisition process by using machine learning techniques (Gallant, 1993; Hassoun, 1995;

Honavar, 1994; Langley, 1995; Mitchell, 1997; Ripley, 1996). A broad range of learning systems have been developed over the years. Our discussion of such systems in this paper has focused mostly on inductive learning systems that learn from examples. Unlike rote learning, inductive learning systems extract, in a relatively compact form (e.g., a few simple classification rules), knowledge that is implicitly provided by a large number of examples. Although different learning algorithms use different mechanisms for limiting the complexity of the concept description learned, they almost always contain a strong bias towards *simple* concept descriptions. This is necessary to avoid overly detailed descriptions that essentially amount to memorization of the training data (with the resulting problem of poor generalization).

Decision tree learning algorithms like ID3, induce classification rules and represent them conveniently as trees. Several decision tree learning algorithms are available in the literature (Quinlan, 1993). They are particularly well-suited for domains in which instances can be described using a finite number of attributes, each of which takes on one of a finite set of values. Extensions that can handle continuous-valued attributes have recently become available. Most decision tree algorithms incorporate a strong search bias in favor of simple trees.

Artificial neural networks (Gallant, 1993; Hassoun, 1995; Ripley, 1996) are particularly well suited for domains in which instances are naturally described using numeric attributes (continuous as well as discrete), or when classification requires consideration of highly nonlinear interactions among several attributes. They are also relatively resistant to attribute noise (noise in the input attributes), as well as classification noise (wrongly classified training examples).

Statistical approaches offer mathematically well-founded techniques for dealing with domains in which knowledge is best represented probabilistically because of uncertainty due to incomplete knowledge, lack of data, or both. Powerful practical algorithms for constructing probabilistic belief networks have begun to appear in the literature (Pearl, 1988; Russell & Norvig, 1995).

Our discussion of inductive learning algorithms has not addressed a number of important theoretical and practical issues that arise in the design and use of such algorithms. The feasibility of inductive learning is determined by various factors including: the size or complexity of the hypothesis space (defined by the concept language), the amount, quality, and form of domain knowledge and training data that is available and the model of learning (e.g., whether examples are chosen randomly or purposefully by the teacher, whether the learning algorithm is allowed to pose queries to the teacher or actively select examples, whether learning has to be done on line while the system is being used to perform classification or it can learn off line, etc.). Detailed exploration of these and related issues is part of much ongoing research in machine learning (Langley, 1995; Mitchell, 1997).

Besides the inductive approaches discussed in this paper, a number of other inductive learning paradigms exist, which we have not examined in this paper because of space constraints. These include: *inductive logic programming* (Lavrac & Dzeroski, 1994) (wherein knowledge is represented the form of logic programs), *automata induction* (Parekh & Honavar, 1997) (wherein knowledge is represented in the form of finite-state automata), etc. In addition, *evolutionary algorithms* (Goldberg, 1989; Holland, 1992; Mitchell, 1996) can be used to search for target concepts in appropriately expressed concept spaces (e.g., decision trees, neural networks, LISP programs, etc.). Examples of evolutionary induction of LISP programs appear in (Koza, 1992), while induction of neural networks is discussed in (Patel & Honavar, 1997).

Apart from inductive learning, deductive or analytical learning algorithms (Langley, 1995; Mitchell, 1997) are also of interest. Essentially, these are knowledge compilation techniques which abstract (or generalize) and store the results of computationally expensive deductive inferences used in solving specific instances of a problem. Once this is done, other similar problems can be solved more efficiently in the future.

All the approaches we have discussed, including automated knowledge acquisition through learning, have assumed that all the attributes used to describe the instances are relevant to the diagnosis task at hand. Practical problems often present an abundance of irrelevant or redundant attributes. In such cases, the learning algorithm has to select a relevant or useful subset of the attributes to use in concept induction. An example of such a scenario which is of significant practical interest is the task of selecting a subset of clinical tests (each with different financial cost, diagnostic value, and associated risk) to be performed as part of a medical diagnosis task. Although several approaches to attribute selection have been developed in the literature, most of them work well only with linear classifiers but not with nonlinear classifiers (e.g., neural networks) (Ripley, 1996). In addition, these techniques fail to deal with multiple selection criteria (e.g., classification accuracy, feature measurement cost, etc.) which might be desirable in many applications (e.g., the medical diagnosis). A recent approach to the attribute-selection problem uses evolutionary algorithms to select a relevant subset of attributes under a variety of cost and performance constraints (Yang & Honavar, 1997).

Our discussion of knowledge acquisition in this paper has assumed that the necessary knowledge and data are available in highly structured formats defined by the instance representation languages used by the individual learning algorithms. The advent of the world wide web and advances in internet technology present scenarios wherein knowledge acquisition might involve knowledge extraction and refinement from a large number of heterogeneous (and possibly geographically distributed) data and knowledge sources. Such sources of diagnostic knowledge might include case repositories residing in multiple databases, textual data (e.g., technical manuals, dictionaries, encyclopedia, etc.), multimedia data (pictures, sounds, etc.), etc. Automated knowledge acquisition from heterogeneous data and knowledge sources is largely an open research problem although some work is currently underway in this area (Fayyad *et al.*, 1996; Honavar & Miller, 1997).

As the preceding discussion indicates, no single AI paradigm adequately meets the diverse design and performance requirements that may need to be met in a broad range of diagnosis applications. This suggests that it might be advantageous to design *hybrid systems* that build on the strengths of multiple AI approaches (Gallant, 1993; Goonatilake & Khebbal, 1995; Honavar & Uhr, 1994; Medsker, 1994; Sun & Bookman, 1994).

For instance, the different AI techniques presented in this paper differ from each other in their choice of search bias, concept representation used (or learned), etc., which results in possibly different classifications of the same input situation. By using a collection of different systems (analogous to multiple experts) and a majority voting scheme to produce the resulting diagnosis, we may hope to perform better than any of the approaches individually (Gallant, 1993; Langley, 1995; Mitchell, 1997; Ripley, 1996).

One might also consider systems that use prior knowledge (e.g., in the form of a model of the domain or diagnostic rules) when available, and use inductive learning to refine and extend the knowledge base. For instance, knowledge available in the form of a decision tree or a set of rules can be encoded into a neural network for further refinement through learning from examples. An example of this approach is offered by (Shavlik, 1994). This approach allows the rules of expert systems to be mapped into neural network architectures, thereby permitting the system to cope with changes in diagnostic knowledge. An added benefit of such a mapping to neural networks (or other suitable parallel computational structures) is the speed-up obtained in producing diagnoses for time-critical applications. Similarly, inductive procedures can be used to succinctly capture large bodies of causal knowledge used in model-based diagnosis. A variety of additional possibilities e.g., new approaches to automated knowledge acquisition from heterogeneous data and knowledge sources, remain to be explored.

References

Buchanan, B. G., & Wilkins, D. C. (eds). (1993). *Readings in Knowledge Acquisition.* Palo Alto, CA: Morgan Kaufmann.

Chen, C-H., & Honavar, V. (1995). A Neural Memory Architecture for Content as Well as Address-Based Storage and Recall. *Connection Science*, **7**, 293–312.

Chen, C-H., & Honavar, V. (1996). A Neural Architecture for High-Speed Database Query Processing. *Microcomputer Applications*, **15**, 7–13.

Dean, T., Allen, J., & Aloimonos, Y. (1995). *Artificial Intelligence - Theory and Practice.* Redwood City, CA: Benjamin/Cummings.

Dempster, A. (1968). A Generalization of Bayesian Inference. *Journal of the Royal Statistical Society*, **30 (Series B)**, 205–247.

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

Durkin, J. (1994). *Expert Systems – Design and Development.* New York, NY: Macmillan Publishing Company.

Fayyad, U., Piatetsky-Shapiro, G., Smyth, P., & Uthurusamy, R. (eds). (1996). *Advances in Knowledge Discovery and Data Mining.* Cambridge, MA. To appear: MIT Press.

Forbus, K. D., & de Kleer, J. (1993). *Building Problem Solvers.* Cambridge, MA: The MIT Press.

Fu, K. S. (1982). *Syntactic Pattern Recognition and Applications.* Englewood Cliffs, NJ: Prentice-Hall.

Fukunaga, K. (1990). *Introduction to Statistical Pattern Recognition.* New York: Academic Press.

Gallant, S. (1993). *Neural Network Learning and Expert Systems.* Cambridge, MA: MIT Press.

Ginsberg, M. (1993). *Essentials of Artificial Intelligence.* San Mateo, CA: Morgan Kaufmann Publishers, Inc.

Goldberg, D. E. (1989). *Algorithms in Search, Optimization and Machine Learning.* New York, NY: Addison-Wesley.

Gonzalez, R. C., & Thomason, M. G. (1978). *Syntactic Pattern Recognition: An Introduction.* Reading, MA: Addison Wesley.

Goonatilake, S., & Khebbal, S. (eds). (1995). *Intelligent Hybrid Systems.* West Sussex: John Wiley.

Hassoun, M. H. (1995). *Fundamentals of Artificial Neural Networks.* Boston, MA: MIT Press.

Hebb, D. O. (1949). *The Organization of Behavior.* New York: Wiley.

Holland, J. (1992). *Adaptation in Natural and Artificial Systems.* Cambridge, MA: MIT Press.

Honavar, V. (1994). Toward Learning Systems That Integrate Different Strategies and Representations. *Pages 561–580 of:* Honavar, V., & Uhr, L. (eds), *Artificial Intelligence and Neural Networks: Steps Toward Principled Integration.* San Diego, CA: Academic Press.

Honavar, V., & Miller, L. (1997). *Automated Knowledge Acquisition from Heterogeneous Distributed Data and Knowledge Sources.* To appear.

Honavar, V., & Uhr, L. (1993). Generative Learning Structures and Processes for Generalized Connectionist Networks. *Information Sciences*, **70**, 75–108.

Honavar, V., & Uhr, L. (eds). (1994). *Artificial Intelligence and Neural Networks – Steps toward Principled Integration.* San Diego, CA: Academic Press.

Hutchinson, A. (1994). *Algorithmic Learning.* New York, NY: Oxford University Press.

Kolodner, J. (1993). *Case-Based Reasoning.* San Mateo, CA: Morgan Kaufmann Publishers, Inc.

Koza, J. (1992). *Genetic Programming.* Cambridge, MA: MIT Press.

Kung, S. Y. (1993). *Digital Neural Networks.* New York: Prentice-Hall.

Langley, P. (1995). *Elements of Machine Learning.* San Mateo, CA: Morgan Kauffman.

Lavrac, N., & Dzeroski, S. (1994). *Inductive Logic Programming Techniques and Applications.* New York: Ellis Horwood.

Luger, G. F., & Stubblefield, W. A. (1993). *Artificial Intelligence.* New York, NY: Benjamin/Cummings.

Medsker, L. (1994). *Hybrid Neural Network and Expert Systems.* New York: Kluwer.

Michalewicz, Z. (1992). *Genetic Algorithms + Data Structures = Evolution Programs.* Berlin: Springer-Verlag.

Michalski, R. S. (1983). Theory and Methodology of Inductive Learning. *In:* Michalski, R. S., Carbonell, J. G., & Mitchell, T. M. (eds), *Machine Learning – An Artificial Intelligence Approach.* Palo Alto, CA: Tioga.

Miclet, L. (1986). *Structural methods in pattern recognition.* New York, NY: Springer-Verlag.

Mitchell, M. (1996). *An Introduction to Genetic Algorithms.* Cambridge, MA: MIT Press.

Mitchell, T. (1997). *Machine Learning.* McGraw Hill. (forthcoming).

Mozetic, I. (1992). Model-Based Diagnosis: An Overview. *Pages 419–430 of:* Marik, V., Stepankova., O., & Trappl, R. (eds), *Advanced Topics in Artificial Intelligence.* Springer-Verlag.

Natarajan, B. (1991). *Machine Learning: A Theoretical Approach.* Palo Alto, CA: Morgan Kaufmann.

Parekh, R.G., & Honavar, V. (1997). Automata Induction, Grammar Inference, and Language Acquisition. *In:* Dale, Moisl, & Somers (eds), *Handbook of Natural Language Processing.* New York, To appear.: Marcel Dekker.

Parekh, R.G., Yang, J., & Honavar, V. (1997). *Multi-Category Constructive Neural Network Learning Algorithms for Real-Valued Pattern Classification.* Ames, Iowa: Tech. Rep. ISU-CS-TR 97-01, Department of Computer Science, Iowa State University.

Patel, M., & Honavar, V. (eds). (1997). *Evolutionary Synthesis of Neural Systems.* Cambridge, MA. To appear.: MIT Press.

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

Puppe, F. (1993). *Systematic Introduction to Expert Systems – Knowledge Representations and Problem Solving Methods.* Berlin: Springer-Verlag.

Quinlan, R. (1993). *C4.5: Programs for Machine Learning.* San Mateo, CA: Morgan Kaufmann.

Rich, E., & Knight, K. (1991). *Aritificial Intelligence.* New York, NY: McGraw-Hill.

Ripley, B. (1996). *Pattern Recognition and Neural Networks.* New York, NY: Cambridge University Press.

Russell, S., & Norvig, P. (1995). *Artificial Intelligence - A Modern Approach.* Englewood Cliffs, NJ: Prentice-Hall.

Schank, R. (1982). *Dynamic Memory: A theory of learning in computers and people.* New York, NY: Cambridge University Press.

Schank, R., & Abelson, R. (1977). *Scripts, plans, goals and understanding.* Northvale, NJ: Erlbaum.

Shafer, G. (1976). *A Mathematical Theory of Evidence.* Princeton, NJ: Princeton University Press.

Shannon, C. (1948). A Mathematical Theory of Communication. *Bell System Technical Journal*, **27**(July and October), 379–423, 623–656.

Shavlik, J. (1994). A Framework for Combining Symbolic and Neural Learning. *Pages 561–580 of:* Honavar, V., & Uhr, L. (eds), *Artificial Intelligence and Neural Networks: Steps Toward Principled Integration.* San Diego, CA: Academic Press.

Shavlik, J., & Dietterich, T. (eds). (1990). *Readings in Machine Learning*. Palo Alto, CA: Morgan Kaufmann.

Simon, H. A. (1983). Why should machines learn? *In:* Michalski, R. S., Carbonell, J. G., & Mitchell, T. M. (eds), *Machine Learning – An Artificial Intelligence Approach*. Palo Alto, CA: Tioga.

Stefik, M. (1995). *Introduction to Knowledge Systems*. Palo Alto, CA: Morgan Kaufmann.

Sun, R., & Bookman, L. (eds). (1994). *Computational Architectures Integrating Symbolic and Neural Processes*. New York: Kluwer.

Tanimoto, S. L. (1995). *Elements of Artificial Intelligence Using Common Lisp*. New York: Computer Science Press.

Uhr, L. (1973). *Pattern Recognition, Learning and Thought*. Englewood Cliffs, NJ: Prentice-Hall.

Winston, P. (1992). *Artificial Intelligence*. New York, NY: Addison-Wesley.

Yang, J., & Honavar, V. (1997). *Feature Subset Selection Using a Genetic Algorithm*. To appear.

Zadeh, L. (1965). Fuzzy Sets. *Information and Control*, **8**, 338–353.

Zimmermann, H. (1991). *Fuzzy Set Theory – And Its Applications*. Second edn. Dordrecht, The Netherlands: Kluwer.