Fault-tolerant incremental diagnosis with limited historical data

SICS Technical Report T2006:17
ISSN 1100-3154, ISRN:SICS-T–2006/17-SE

2006-11-28

Daniel Gillblad[1], Anders Holst[1], and Rebecca Steinert[1]

Swedish Institute of Computer Science, Box 1263, SE-164 29 Kista, Sweden
{dgi, aho, rebste}@sics.se

Abstract. In many diagnosis situations it is desirable to perform a classification in an iterative and interactive manner. All relevant information may not be available initially and must be acquired manually or at a cost. The matter is often complicated by very limited amounts of knowledge and examples when a new system to be diagnosed is initially brought into use. Here, we will describe how to create an incremental classification system based on a statistical model that is trained from empirical data, and show how the limited available background information can still be used initially for a functioning diagnosis system.

1 Introduction

Real world diagnosis is often complicated by the fact that all relevant information is not directly available. In many diagnosis situations it is impossible or inconvenient to find all input values to a classifier before being able to make a classification, and we would therefore like the classifier to act as an interactive decision support system that will guide the user to a useful diagnosis.

A typical example of this could be the diagnosis of faults in a vehicle that needs servicing. All relevant information for a correct diagnose is not available when the vehicle arrives for service, but must be acquired manually. This procedure is often very time consuming, involving inspection of parts and systems that can be difficult to get at and to evaluate. It may also require much experience and training to know what to look for. The scenario is similar in e.g. many cases of medical diagnosis, where some information, perhaps the results of certain lab tests, must be acquired manually at a cost that can be measured in both monetary terms and in terms of a patients well-being.

To deal with these issues, we have constructed an incremental diagnosis system that is trained from empirical data. During a diagnosis session it evaluates the available information, calculates what additional information that would be most helpful for the diagnose, and asks the user to acquire that information. The process is repeated until a reliable diagnose can be performed. This system both

significantly speeds up the diagnosis and represents knowledge of best practises in a structured manner. Since knowledge about how to perform efficient diagnostics of this kind of systems often is acquired in time by people working with it, the diagnose system is potentially very helpful for preserving diagnostics capabilities when personnel needs to be changed and to increase the diagnostic capabilities of novice users [8].

However, the matter is complicated by the fact that we often have limited amounts of background knowledge and examples when a new system to be diagnosed is initially brought into use. We show how the limited available background information can still be used initially for a functioning diagnosis system, until sufficiently many new examples of real diagnosis has been collected from the use of the system. The cost and complexity for the user of acquiring certain information must also be taken into account, and the system should e.g. refrain from asking about information that is very time consuming to acquire until it is absolutely necessary. Other complicating factors include that many errors arise because of misuse or wrongly setup equipment, and incorrect answers and inputs to the diagnosis system.

We will here present a system for incremental diagnosis where the issues above are addressed.

2 Practical incremental diagnosis

Using computers to support the diagnostic process is one of the classical applications of artificial intelligence. The methods developed so far are usually expert-systems related and rule-based [17, 7], neural network related [6, 9], or based on probabilistic methods [18, 5, 11]. The rule-based systems are in essence implementations of diagnostic protocols, specified in a large part manually through expert knowledge and to a lesser degree by learning from examples.

The approach works well in many diagnostic situations, especially in areas where expert knowledge is easily extracted and where the system does not need to adapt to new classification examples. However, rule based systems often suffer from a very rigid decision structure, where questions have to be asked in a certain order reflecting the internal representation. This might be very problematic in practise, where the order of which data can be retrieved is often arbitrary. Rule-based systems also suffer from the complexity of the rules that need to be implemented, resulting in a high number of conditions that need to be described, and from problems in dealing with uncertain sources of evidence. These issues can be solved, at least to a certain degree, by basing the diagnosis system on a probabilistic model.

Creating a useful probabilistic diagnosis system may not be overly complicated, but there are a few considerations and requirements worth keeping in mind. First, while being robust to erroneous and uncertain inputs, it is important that the number of questions necessary to reach a classification is minimised. In practise, this demand must usually be formulated somewhat differently in

that we actually want to minimise the total *cost* of acquiring the information necessary to reach a classification, not just the number of questions.

No matter what methods that are used in the diagnosis system, we have to decide whether to base the system on available expert knowledge, on examples from earlier diagnosis situations using a more data-driven approach, or on both. If we want to be able to update the system as new examples are classified, a data-driven approach is preferable. However, as we mentioned earlier, quite often there are no examples of earlier diagnoses available when a new system is taken into use. This means that we have no choice but to try to incorporate some expert knowledge into the system so that it is at least somewhat useful at its introduction.

Although it is quite possible to encode expert knowledge into the structure of a probabilistic model, such as a *Bayesian Belief network* [16, 15], we will use a slightly different approach by representing this expert knowledge as a special form of examples. If we refer to a regular classification example as a *case*, these special form of examples, *prototypes*, can be seen as generalisations of typical cases [10]. Each prototype represents a *typical* input vector for a certain class. A number of prototypes can be used to efficiently encode prior knowledge about a system. However, as cases and prototypes do have rather different interpretations, the diagnostic system system must be able to account for this difference.

The approach also allows us to let the diagnostic system act as a persistent repository of knowledge that can continuously incorporate information from new classification cases, which may be difficult in rule-based systems. As the system gathers more information, some of the acquired cases may also be generalised by an expert into prototypical data, which could be highly beneficial for both classification performance and system understanding.

Another very practical requirement on the diagnostic system is that the probabilistic model in many cases must be able to handle a mixture of both continuous and discrete inputs. This complicates matters somewhat, as we will see in later in the description of the models, especially if there are constraints on the time the system can use to calculate which unknown attribute to ask a question on. We also have to consider the fact that some attributes are grouped, in the sense that they are all acquired at the same time, by the same measurement. This complicates question generation somewhat, but can be taken care of within the model.

3 Probabilistic methods for query-reply systems

As an alternative to rigid decision tree models for incremental diagnosis, we can use a probabilistic model [4, 14, 13]. Instead of modelling the decision tree directly, we model the relations between input data and the classes by describing them in a probability model. The probability of a class Z given a set of inputs $\boldsymbol{X} = X_1, X_2, \ldots, X_n$ can be written as

$$p(Z|\boldsymbol{X}) = \frac{p(Z)p(\boldsymbol{X}|Z)}{p(\boldsymbol{X})} \tag{1}$$

Depending on the form of the conditional distributions, estimation of these distributions and calculation of the above expression can be very difficult in practise. A very common simplifying assumption is to assume conditional independence between all input attributes $\boldsymbol{X}$ given the class attribute Z. The complete distribution can then be described as a product of the probabilities of the individual attributes,

$$p(Z|\boldsymbol{X}) \propto p(Z)p(\boldsymbol{X}|Z) = p(Z)\prod_{i=1}^{n} p(X_i|Z) \tag{2}$$

The distribution for each attribute given a specific class, $P(X_i|Z)$ is significantly easier to estimate, due to the relatively low number of free parameters. This model is usually referred to as a *Naive Bayesian classifier* [3]. It often gives surprisingly good classification results although the independence assumption is usually only partially fulfilled, and by its simplicity very suitable as a starting point for statistical incremental diagnosis systems.

Creating an incremental diagnosis system, or a *query-reply system*, based on a statistical model such as the Naive Bayesian Classifier is relatively straightforward. Essentially, we would like to perform three operations on the model. The first is to find the class distribution if we know the values of a number of input attributes. This being an inherent property of a classification model, we will assume we can perform this. Second, we would like to calculate how much each unknown input attribute is likely to contribute to the classification given a number of known attributes. Finally, we would also like to be able to estimate how much each known input attribute contributed to a classification to provide feedback to the user. First, let us have a look at how we this can be formulated without considering the details of the statistical model.

To determine the probable impact of gaining information about a currently unknown attribute, we can calculate the *expected reduction in entropy* in the class distribution if we learn the value of the attribute. If Z is the class distribution, $\boldsymbol{x} = X_1 = x_1, \ldots, X_n = x_n$ the already known input attributes, and Y the unknown attribute we want to calculate the impact of, we can write this *entropy gain* $G_H(Y)$ as

$$G_H(Y) = H(Z|\boldsymbol{x}) - E_Y[H(Z|\boldsymbol{x}, Y)] \tag{3}$$

where $H(X)$ denotes the entropy [20] of a stochastic variable X and E the expectation with regards to Y. This expression is guaranteed to be equal to or larger than zero since conditioning on the average reduces entropy. If the unknown attribute Y is discrete, the expression becomes

$$G_H(Y) = H(Z|\boldsymbol{x}) - \sum_{y_j \in Y} p(Y = y_j|\boldsymbol{x})H(Z|\boldsymbol{x}, Y = y_j) \tag{4}$$

There is no requirement that Y must be discrete. However, calculation of the expectation in equation 3 may require numerical calculation of a rather complicated integral. The expression can in some situations be estimated with limited computational effort. If not, as long we can draw random numbers from $p(Y|\boldsymbol{x})$, we can always resort to calculating the expression through using a Monte-Carlo

approach, where we draw N samples y_i from $p(Y|\boldsymbol{x})$ and approximate the entropy gain by

$$G_H(Y) = H(Z|\boldsymbol{x}) - \frac{\sum_{j=1}^{N} p(Y = y_j|\boldsymbol{x}) H(Z|\boldsymbol{x}, Y = y_j)}{\sum_{j=0}^{N} p(Y = y_j|\boldsymbol{x})} \tag{5}$$

This expression usually converges quite fast, but may still require prohibitively extensive computational resources. Also note that for none of these expressions there is in fact no restriction on the class attribute Z to be discrete as long as we are able to effectively calculate the entropy of the distribution.

Accounting for the costs associated with performing the queries is very straightforward as long as they are all measured at the same scale. If the cost of an unknown attribute is $C(Y)$, we can calculate the cost weighted entropy gain G_{H_C} as

$$G_{H_C} = C(Y) G_H(Y) \tag{6}$$

This can then be used to rank attributes for query instead of the entropy gain. The expression is similar, but not equal to what is often referred to as the *value of information*, usually defined as the expected reduction in cost compared with making a diagnose without the information [19].

Also note that in practical applications we may need to consider attributes to be grouped, in the sense that their values are acquired together. As an example, this could be a number of values that are provided by a time-consuming lab test. In this case, the cost of acquiring all these values must be weighed against the expected reduction in entropy of acquiring *all* the attributes,

$$G_H(Y) = H(Z|\boldsymbol{x}) - E[H(Z|\boldsymbol{x}, \boldsymbol{Y})] \tag{7}$$

where $\boldsymbol{Y}$ represents these grouped unknown attributes. Calculation of this expression may however be very time consuming in the case $\boldsymbol{Y}$ contains many attributes.

In some situations, the user of the diagnose system might have a relatively good idea of what type of class the output should actually be, and would like to know how much each unknown attribute is expected to contribute to determining whether it actually is this class or not. This can be calculated in a manner similar to expression 3, but instead we calculate the expected magnitude of change in probability of the class being z, $G_p(Y)$:

$$G_p(Y) = |p(Z = z|\boldsymbol{x}) - E[p(Z = z|\boldsymbol{x}, Y)])| \tag{8}$$

That is, the absolute value of the difference in probability of class z with and without knowing attribute Y. However, this calculation is not necessarily useful in all diagnosis situations.

To estimate the *explanation value* of a certain known attribute, i.e. how much knowing the value of this attribute contributes to the classification, we can simply calculate the difference in entropy of the class Z when the attribute is known

and when it is not. If Y is the class attribute we would like to calculate the explanation value of and $\boldsymbol{X}$ all other known attributes as above, the explanation value $E_H(Y)$ is

$$E_H(Y) = H(Z|\boldsymbol{x}) - H(Z|\boldsymbol{x}, Y) \tag{9}$$

This expression may not perfectly reflect the contribution to the classification of a certain attribute if there are dependencies between the input attributes, but should be perfectly usable in most practical situations.

Related to expression 8, the user may want to know how much each known attribute contributed to the certainty of the class having a specific outcome. This can be easily calculated as

$$E_p(Y) = |p(Z = z|\boldsymbol{x}) - p(Z = z|\boldsymbol{x}, Y)| \tag{10}$$

that is the absolute value of the change in probability of class z knowing attribute Y or not.

It is worth noting that although the above expressions are conceptually simple, classification can easily get computationally intractable with some models for large domains [12]. However, it is quite possible to construct a model that is simple enough to be computationally usable but still addresses the issues we have discussed so far.

4 Incremental diagnosis with limited amounts of examples

Let us now construct a statistical model for incremental diagnosis that can effectively make use of limited historical data. Although using a statistical model, we are here going to take an approach that is not that different from an instance based learner, where each new pattern is compared to all examples in the available historical data to find the most similar ones [1]. Assuming a clear distinction between prototypical and case data, the former describing a typical, distilled instance of a class, and the latter an example of a diagnostic case, we let each prototype form the basis of a component in a *mixture* [2], a weighted sum, of Naive Bayes classifiers. That is, if there are m prototypes in the data, we create a mixture of m simpler classifiers, whose classifications are then combined. This also allows us to effectively describe classes that manifest themselves in rather different manners in data, as long as there are available prototypes that roughly describe these different situations.

More formally, if $\boldsymbol{X} = X_1, \ldots, X_n$ denote the input attributes, the class distribution Z can be written as

$$p(Z|\boldsymbol{X}) \propto \sum_{z \in Z} p(Z = z) \sum_{k \in P_z} \left(\pi_{z,k} \prod_{i=1}^{n} p_{z,k}(X_i|Z = z) \right) \tag{11}$$

where P_z is the set of prototypes that are labelled with class z, and $\pi_{z,k}$ denotes the mixing proportion corresponding to prototype k for class z. To arrive at the actual distribution $p(Z|\boldsymbol{X})$, we only need to normalise over Z in equation 11.

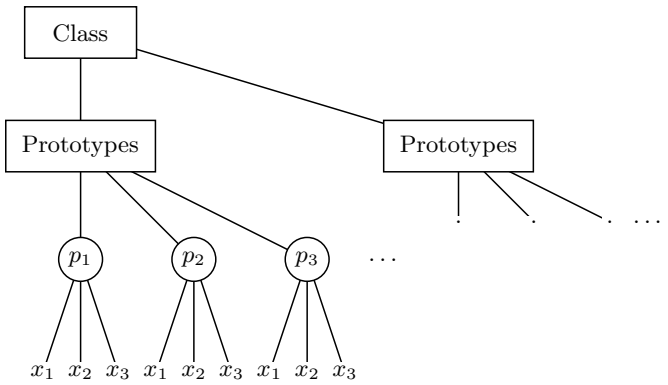

Fig. 1. An overview of the model structure. Each class is associated with a number of prototypes, $p_1, p_2, \ldots$. The prototypes all use the same input attributes $x_1, x_2, \ldots$, each prototype corresponding to a single prototypical entry in the historical data.

In equation 11, we still need to describe, and eventually estimate, both the prior class distribution, $p(Z)$, and all conditional distributions for each attribute, prototype, and class, $p_{z,k}(X_i|Z = z)$. These conditionals distributions are assumed to be either discrete, in the case X_i represents a nominal valued attribute, or Gaussian (normal) in the case of continuous attributes,

$$p_{z,k}(X_i|Z = z) = \frac{1}{\sqrt{(2\pi)\Sigma_i}} \exp\left(-\frac{(x_i - \mu_i)^2}{2\Sigma_i}\right) \tag{12}$$

where μ is the mean and Σ the variance of the distribution. Assuming Gaussian distributions for continuous attributes usually works well in practise even if the actual generating distributions are rather different. Gaussian distributions can however be avoided by discretising the continuous attributes, which in some cases actually may be beneficial both for classification precision and computational complexity. If an attribute is known to operate within certain distinct regions, e.g. a temperature that can be considered low, normal, or high, discretising it to these classes might be useful. However, in most other cases the use of a Gaussian distribution is more straightforward.

Now that we have decided what parameterisations to use, we need to be able to estimate these parameters from data. Let us start with the mixing proportions $\pi_{z,k}$. These essentially represent the relative importance of each prototype within a class, where $\sum_{k \in P_z} \pi_{z,k} = 1$. They are set manually, and should roughly correspond to the proportion of actual diagnosis cases that the prototype usually represents within the class. However, as this is not necessarily known, they can for all practical purposes be set to equal importance as $1/m_z$, where m_z is the number of prototypes for class z.

To arrive at an effective estimation procedure for the prior class distribution $p(Z)$ and the conditional attribute distributions $p_{z,k}(X_i|Z = z)$, we are going to use a Bayesian approach [3]. That is, we are going to assume that the parameters of the distributions are themselves stochastic variables with *prior distributions*. We are however not going to use a fully Bayesian approach for the whole class distribution in equation 11, but only for the class prior and the mixture component distributions. We will also not present complete derivations of the expressions used, but would like to refer the reader to [22] where they are carried out in detail.

To be able to properly incorporate the prototypical data with the actual cases, we are going to use a hierarchy of priors that will be estimated in turn before we arrive at the final distribution. First, we will assume a non-informative uniform prior, that is used to estimate a distribution from the prototype data. This distribution in turn will, in a sense, be used as a prior for the estimation of the actual distribution. Let us start with describing how this is performed for the discrete distributions $p(Z)$ and $p_{z,k}(X_i|Z = z)$.

In the case of the discrete class distribution Z, the parameter p_z representing the probability of each outcome z can be estimated through

$$p_z = \frac{c_z^c + \eta \left(\frac{c_z^p + \theta}{C^p + \theta |Z|} \right)}{C^c + \eta} \tag{13}$$

where c_z^p and c_z^c are the number of outcomes z in the prototype and case data respectively, and C^p and C^c the total number of examples in each of these data sets. $|Z|$ denotes the total number of outcomes in Z. The parameter η represents how much trust we put in the prior distribution estimated from the prototypes, while η can be interpreted as a kind of smoothing parameter for the prior distribution.

To be able to properly incorporate the prototypical data with the actual cases, each conditional $p_{k,z}(X_i|Z = z)$ for a certain class and prototype is estimated from corresponding cases using a prior distribution, in turn estimated from the specific prototype. This estimation from a specific prototype uses a prior estimated from all prototypical data, which in turn uses a non-informative uniform prior. Let us walk through these estimations step by step, starting with the estimation of a prior distribution based on all prototypes.

In the case $p_{k,z}(X_i|Z = z)$ is discrete, the parameter p_x^p representing the probability of each outcome X is estimated through

$$p_x^p = \frac{c_x^p + 1}{C^p + |X|} \tag{14}$$

where c_x^p are the number of outcomes x in the data, C^p the total number of prototypes, and $|X|$ the number of outcomes in X. In the continuous case, we

estimate the parameters of a Gaussian as

$$\mu^p = \sum_{\gamma \in D_p} x^{(\gamma)} / C^p \tag{15}$$

$$\Sigma^p = \sum_{\gamma \in D_p} (x^{(\gamma)} - \mu^p)^2 / (C^p - 1) \tag{16}$$

where D_p represents the set of prototypes and $x^{(\gamma)}$ one prototype value.

Now, before we incorporate the case data, we will estimate the distribution $p^0_{z,k}(X_i|Z = z)$, which represents the final parameter estimation in case there is no case data available, and otherwise forms the basis of the parameter estimation from case data. In the discrete case the parameter p^0_x is estimated as

$$p^0_x = \frac{v^k_x + p^p_x / |D_p|}{1 + 1/|D_p|} \tag{17}$$

where v^k_x is an indicator variable that is one if the outcome is equal to x and zero otherwise, and $|D_p|$ the number of prototypes in the data. In the continuous case, the parameters are estimated as

$$\mu^0 = \frac{x_k + \mu^p / |D_p|}{1 + 1/|D_p|} \tag{18}$$

$$\Sigma^0 = \Sigma^p \tag{19}$$

where x_k represents the value of X in prototype k. The mean of the distributions varies with the prototype, while the variance is the same for all of them.

To make efficient use of the case data, we want to use it for estimation in such a way that each prototype distribution is updated in proportion to how likely it is that each specific case was generated from it. In detail, the importance of each case for a certain prototype k is weighted by the probability that the case was generated from it by calculating the likelihood that each case was generated from prototype k and normalising over the patterns within the class,

$$p(k|\boldsymbol{x}) = \frac{\pi_k \prod_{i=1}^n p_{z,k}(x_i|Z = z)}{\sum_j \pi_j \prod_{i=1}^n p_{z,k}(x_j|Z = z)} \tag{20}$$

where $\boldsymbol{x}$ denotes a case pattern. The procedure can be viewed as performing one step of the *Expectation Maximisation* algorithm for the mixture. The final expressions for the parameters for a certain prototype k in the discrete case then becomes

$$p_x = \frac{\sum_{\gamma \in D_c} p(k|\boldsymbol{x}^{(\gamma)}) v^{(\gamma)}_x + \psi p^0_x}{\sum_{\gamma \in D_c} p(k|\boldsymbol{x}^{(\gamma)}) + \psi} \tag{21}$$

where $v^{(\gamma)}_x$ is an indicator variable that is one if $X = x$ and zero otherwise. D_c denotes the set of cases and $\boldsymbol{x}^{(\gamma)}$ case γ in this set. In the continuous case, the

parameters are estimated through

$$c_k = \sum_{\gamma \in D_c} p(k|\boldsymbol{x}^{(\gamma)}) \tag{22}$$

$$\mu^c = \frac{\sum_{\gamma \in D_c} p(k|\boldsymbol{x}^{(\gamma)}) x^{(\gamma)}}{c_k} \tag{23}$$

$$\Sigma^c = \frac{\sum_{\gamma \in D_c} p(k|\boldsymbol{x}^{(\gamma)}) (x^{(\gamma)} - \mu^c)^2}{c_k} \tag{24}$$

$$\mu = \frac{c_k \mu^c + \psi \mu^p}{c_k + \psi} \tag{25}$$

$$\Sigma = \frac{c_k \Sigma^c + \psi \Sigma^p + \frac{c_k \psi}{c_k + \psi} (\mu^c - \mu^p)^2}{c_k + \alpha} \tag{26}$$

where $x^{(\gamma)}$ is case value γ, and μ and Σ the final parameter estimates. In both the discrete and continuous case, the parameter ψ represents how much trust we put in the prototypes compared to the cases and can be expected to be set to different values for different applications.

The parameter estimations described above may seem convoluted at first, but are in fact easily and efficiently calculated from available data. Admittedly, the whole estimation procedure is somewhat of an abuse of Bayesian methodology, but one that works very well in practise in terms of regularisation.

Calculating the entropy gain as given by expression 3 and 4 is straightforward, as $p(Y = y_j|\boldsymbol{X})$ can be directly calculated from

$$p(Y|\boldsymbol{X}) \propto \sum_{z \in Z} p(Z = z|\boldsymbol{X}) \sum_{k \in P_z} \pi_k p_k(Z) p_k(Y|Z) \tag{27}$$

where $p(Z|\boldsymbol{X})$ is calculated from equation 11 using the known input attributes. If there are continuous attributes represented by Gaussians present, the situation is a little different, as we have to integrate over the attribute in question instead of calculating the sum in equation 4. This can be solved by using a Monte-Carlo approach, where a number of samples are generated from $p(Y = y_j|\boldsymbol{X})$ and used to calculate the expectation. Another solution that produces very exact results is to sample a number of points from each prototype model, e.g. at equal intervals up to a number of standard deviations from the mean. This can, however, lead to a quite high computational complexity when there is a large number of prototypes in the data.

4.1 Experiments

When a diagnosis is performed, unknown attributes are incrementally set based on the largest entropy gain. Thus, unknown attributes are set in the order in which their contribution to the final *hypothesis* is maximised. The order in which different attributes are set depends mainly on the significance ψ between prototypes and cases. In addition, the significance η of the class distribution may also

influence the importance of specific attributes. Often only a few significant attributes need to be known in order to obtain a final hypothesis, while remaining attributes are redundant in the current context.

While attributes are not necessarily set strictly based on the entropy gain in a real-world diagnosis scenario, since other factors may influence the choice of attribute to set for the user, for testing purposes we will assume that they are. Also, in a real world diagnosis scenario it is not necessarily important to determine at what exact point in the answering sequence we should claim that we have a valid hypothesis about the class. The user can usually determine this reliably just by looking at a presentation of the class distribution, which also provides information on uncertainty and alternative hypothesis. However, being able to signal to the user that we have a valid hypothesis is naturally useful, and for testing absolutely necessary as we need to be able to automatically decide when we have a relevant classification.

A natural way of determining when we have reached a hypothesis is to see if one class has significantly higher probability than the other classes. Unfortunately, it is by no means easy to give a general expression for what constitutes a "significantly higher" probability. The measure is also flawed in that it in practise often is impossible to find one class with significantly higher probability, e.g. if two classes are expressed in almost exactly the same way in the data. Instead, we can rely on the calculated entropy gain for the unknown attributes. To reduce the number of attributes that do not significantly contribute to the actual hypothesis, we have introduced a thresholding parameter ξ. When the entropy gain for all unknown attributes is smaller than ξ, the hypothesis is considered final. This is an unbiased measure of the validity of the hypothesis that does not suffer from the problems discussed above.

In order to test the general performance of our model, we have performed several series of experiments using datasets that contain discrete or continuous attributes. In all of the experiments, we have measured the number of questions needed to achieve a final hypothesis and the number of correctly diagnosed samples. In order to reduce computational demands, the results of the diagnostic performance are based on the average of no more than 10 runs to obtain statistical significance in all of the experiments. Further, in some cases two (or more) classes in the datasets contain nearly similar sets of known attributes, which leads to ambiguous diagnoses. In order to avoid such diagnoses, we allowed for the diagnostic model to use a first and second trial, in which the first and second diagnoses of highest confidence were tested against the target diagnosis.

Experiments on discrete datasets We have here tested the diagnostic performance of our model when varying ξ, ψ and η. For this purpose, three original datasets with discrete attributes were used. The first dataset contains 32 classes of animals described by a total of 82 attributes. The second dataset contains 31 classes of common mechanical faults that appears in military terrain vehicles, described by 39 attributes. The third dataset contains 18 classes of common mechanical faults that appears in military tanks, described by 83 attributes. While

the first data set is artificial, the other data sets represent real data on which the model will be used in practise.

These original datasets are completely clean, in the sense that they do not contain any unknown attributes, or, as the data from a testing viewpoint also must be interpreted as the true solutions, any incorrect attributes. In order to evaluate our complete diagnostic model, synthetic prototypes and cases were extracted from the original datasets. Prototypes and cases are here defined to contain both known and unknown values. In addition, cases can also contain incorrect attribute values.

For each class, two prototypical samples were extracted directly from the original datasets. Based on the complementary prior for each attribute, the value was set as unknown or to the correct known value. The prototypes were then used in the diagnostic model to create five synthetic cases for each class. Based on the largest entropy gain, specific attributes were set using known values from the original dataset. As mentioned earlier, cases can contain incorrect attribute values. Therefore, noise was introduced through setting 20% of the known attributes in each case to a value based on the prior for the specific attribute.

Initially, two series of baseline experiments for each dataset were performed while varying ξ. In both experiments, each one of the datasets were used as prototypes. In the first series of experiments, the same dataset was used to directly set the attribute values without noise. In the second series of experiments, a subset of the attributes (20%) were randomly selected to contain noise as described, while remaining attribute values were set directly from the original dataset. In practise 20% of noise usually results in around 10% incorrect values among the known attributes during a diagnostic session.

	Without noise			With 20% noise		
ξ	Correct (%) Trial #1	Correct (%) Trial #2	Known attributes	Correct (%) Trial #1	Correct (%) Trial #2	Known attributes
4.7×10^{0}	3.13	3.13	0	3.13	3.13	0
4.7×10^{-1}	100.00	0.00	5.03	56.56	8.75	5.08
4.7×10^{-2}	100.00	0.00	5.69	95.00	1.88	8.50
4.7×10^{-3}	100.00	0.00	6.34	96.25	1.56	9.96
4.7×10^{-4}	100.00	0.00	7.19	97.50	0.94	10.69
4.7×10^{-5}	100.00	0.00	7.91	98.44	1.25	13.03
4.7×10^{-6}	100.00	0.00	8.66	99.06	0.94	13.81

Table 1. Animal dataset. The table shows the average diagnostic performance achieved in one of two trials and the number of attributes needed to obtain a final hypothesis when varying ξ.

When performing diagnoses in which all known attribute values are correct, we observe from the results in tab. 1-2 that the number of known attributes

	Without noise			With 20% noise		
ξ	Correct (%) Trial #1	Correct (%) Trial #2	Known attributes	Correct (%) Trial #1	Correct (%) Trial #2	Known attributes
4.7×10^{0}	3.23	3.23	0.0	3.23	3.23	0.0
4.7×10^{-1}	87.10	12.90	4.81	62.58	15.16	5.67
4.7×10^{-2}	87.10	12.90	5.55	75.48	14.19	7.72
4.7×10^{-3}	87.10	12.90	6.10	80.32	13.23	8.42
4.7×10^{-4}	87.10	12.90	7.58	78.07	15.48	10.64
4.7×10^{-5}	87.10	12.90	8.39	79.03	17.74	11.95
4.7×10^{-6}	87.10	12.90	9.81	80.97	14.84	13.77

Table 2. Terrain vehicle dataset. The table shows the average diagnostic performance achieved in one of two trials and the number of attributes needed to obtain a final hypothesis when varying ξ.

needed to obtain a final hypothesis is close to $\log_2 n$, where n is the total number of attributes in the dataset. When 20% noise is used, we observe that ξ can be used to improve the diagnostic performance on the first trial. Thus, the need for explicit inconsistency checks for incorrect attribute values is reduced, since our diagnostic model is able to find the correct diagnosis, using only a few more known attributes. Further, if the diagnostic model is allowed to use a second trial, we observe that it is possible to obtain a total diagnostic performance of more than 95% in all of the baseline experiments.

	Without noise			With 20% noise		
ξ	Correct (%) Trial #1	Correct (%) Trial #2	Known attributes	Correct (%) Trial #1	Correct (%) Trial #2	Known attributes
4.7×10^{0}	5.56	5.56	0.00	5.56	5.56	0.00
4.7×10^{-1}	100.00	0.0	4.28	65.56	7.22	4.28
4.7×10^{-2}	100.00	0.0	5.17	93.33	2.78	7.46
4.7×10^{-3}	100.00	0.0	6.11	97.78	1.11	8.89
4.7×10^{-4}	100.00	0.0	7.39	98.33	1.11	10.46
4.7×10^{-5}	100.00	0.0	8.17	99.44	0.56	11.39
4.7×10^{-6}	100.00	0.0	9.39	98.33	1.11	12.90

Table 3. Tank dataset. The table shows the average diagnostic performance achieved in one of two trials and the number of attributes needed to obtain a final hypothesis when varying ξ.

Similar to the baseline experiments, two additional series of experiments for each one of the datasets were performed using extracted prototypes and cases. In these experiments, diagnoses were performed when varying the parameters ψ and η while keeping $\xi = 4.7 \times 10^{-5}$ fixed. In fig. 2-4, the diagnostic performance on each dataset and the number of known attributes needed for a final hypothesis is shown. We observe that the significance of prototypes in general needs to be large in order to obtain a diagnostic performance closer to the baseline experiments. In practise this is a feasible result since access to large sets of cases is limited in the initial stages of training. Since the prototype dataset usually contains clean samples, there is no particular reason to lower the significance of prototype data until large amounts of cases have been collected. Further, we observe that the number of known attributes needed to obtain a final hypothesis decreases when when ψ is large. Since all classes are uniformly distributed, the overall diagnostic performance does not change significantly when η is varied.

Experiments on continuous data We have also performed experiments using a dataset that contains both continuous and discrete attributes. Samples in the dataset were extracted from the evaporation stage of a paper mill. The dataset contains 11 classes with 6 samples each, specified by 106 continuous attributes and 4 discrete attributes. Since prototypical samples were unavailable, we used a synthetic set of prototypical samples with all attribute values set to unknown, while using the whole dataset as cases in the model. A fixed subset of the dataset was used to set attribute values when diagnostics was performed.

Three series of experiments were performed in which we varied ψ and ξ. Since the classes are uniformly distributed we did not perform any experiments varying η. In the first series of experiments, the attribute values were set directly without induced noise. In the two remaining series of experiments, we used two different approaches to induce 20% noise in order to investigate how noise-sensitive the model is when using continuous data. Thus, in the second series of experiments, the probability of noise was based on a Gaussian prior distribution estimated for each attribute, calculated from the sample mean and standard deviation measured within each class. In the third series of experiments, the Gaussian prior for each attribute was estimated from the sample mean and standard deviation, measured on the whole dataset.

We observe from the first series of experiments that the diagnostic performance mainly is dependent on the value of ψ (fig. 5a). Naturally a small value on ψ leads to a higher classification rate since the model is based only on real cases. We observe that the total average classification rate can be improved combining the result of two trials (fig. 5a, b). In fig. 5b the classification rates varies compared to the results in fig. 5b. It is likely that the classification rates in the second trial is more susceptible to the parameter settings and possibly to the Monte-Carlo sampling step that is performed on continuous data. Further, our results indicate that the number of steps needed for a final hypothesis depends more on ψ than on ξ, specifically when ψ is very large or very small. For an average of 7.03 known attributes (taken over all classes and 10 runs) we obtain a classification

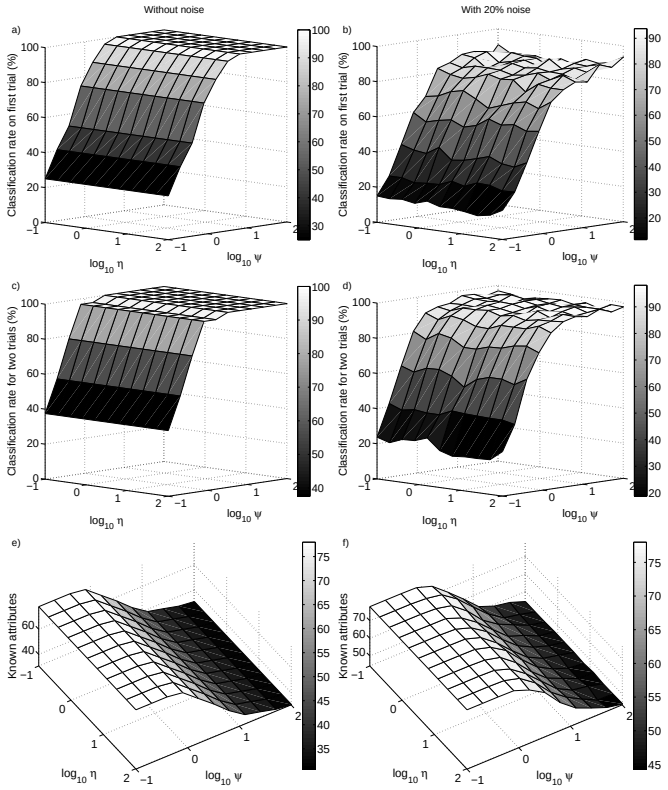

Fig. 2. Average diagnostic performance on the animals dataset when varying η and ψ. The figure shows a) the classification rate on the first trial, b) the classification rate on the first trial with 20% noise, c) the combined classification rate for two trials, d) the combined classification rate for two trials with 20% noise, e) the number of known attributes needed for a final hypothesis and f) the same as in e) but with 20% noise.

rate of 100% using $\psi = 0.001$ and $\xi = 10e^{-5}$, compared to 4.76 known attributes using $\xi = 0.01$ (fig. 5a,c). However, when ψ is set to a fixed medium value, we observe that the model produces nearly the same classification rate using much fewer steps when increasing ξ (fig. 5c). In this case increasing the value on ψ reduces the separability between classes. In effect, the significance of varying ξ is increased, such that it affects only the number of known attributes needed for a final hypothesis but not necessarily the classification rate. However, this only applies as long as ψ is set to a medium value. When ψ is sufficiently large the final hypotheses starts to repeatedly indicate only a limited (and possibly fixed) subset of the classes which reduces the classification rate and the significance of ξ. In this case, for instance, the final hypotheses indicate only one certain class when ψ is sufficiently large, regardless the value of ξ. Conversely, when ψ is set

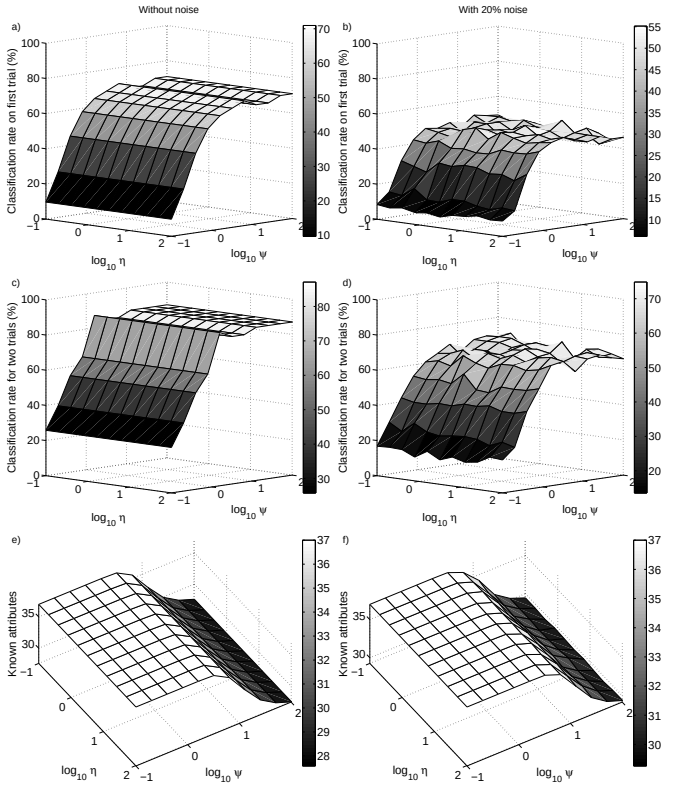

Fig. 3. Average diagnostic performance on the terrain vehicle dataset when varying η and ψ. The figure shows a) the classification rate on the first trial, b) the classification rate on the first trial with 20% noise, c) the combined classification rate for two trials, d) the combined classification rate for two trials with 20% noise, e) the number of known attributes needed for a final hypothesis and f) the same as in e) but with 20% noise.

to a very small value, class separability is increased and the significance of ξ is reduced.

In the second series of experiments, we observe that the diagnostic performance slightly decreases compared to the first series of experiments (fig. 6a, b). Our results indicate that by inducing noise based on the prior Gaussian distribution for each attribute within each class, we can obtain satisfactory performance if ψ is set low. In the third series of experiments, we observe that the diagnostic performance is less robust to noise (fig. 7a, b), compared to the results in the second series of experiments. For example, the number of correct diagnoses is 70.91% compared to 100% in the second series of experiments when $\psi = 10^{-5}$ and $\xi = 10^{-5}$. Further, we observe that the number of known attributes needed

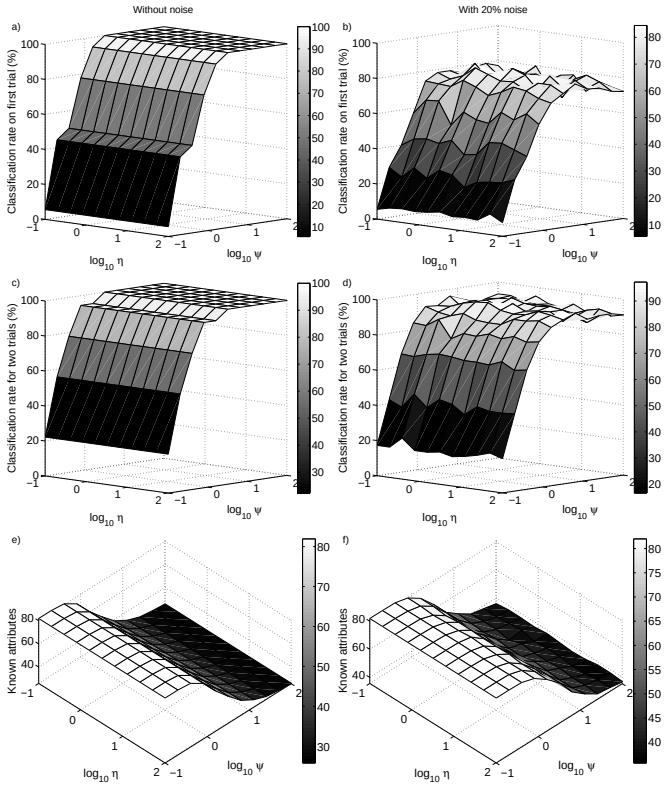

Fig. 4. Average diagnostic performance on the tank dataset when varying η and ψ. The figure shows a) the classification rate on the first trial, b) the classification rate on the first trial with 20% noise, c) the combined classification rate for two trials, d) the combined classification rate for two trials with 20% noise, e) the number of known attributes needed for a final hypothesis and f) the same as in e) but with 20% noise.

for a final hypothesis varies depending on how noise is induced (fig. 5c, 6c and 7c).

Obtained classification rates indicate that the model is more sensitive when noise, based on the Gaussian prior distribution for each attribute estimated over the whole dataset, is induced. It should be noted that the attribute values have a large variance between different classes which therefore causes a lower diagnostic performance when noise is induced this way. We therefore conclude that the model is more robust for inexact attribute values as long as the values are set within the prior distribution of the class.

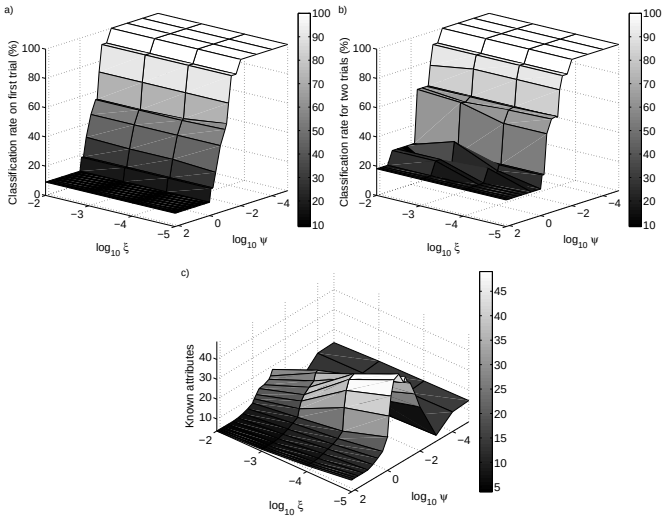

Fig. 5. Diagnostic performance without noise on the evaporation dataset when varying ξ and ψ. The figure shows a) the classification rate on the first trial, b) the combined classification rate for two trials and c) the number of known attributes needed for a final hypothesis.

5 Anomalies, inconsistencies, and settings

In practise, incremental diagnosis poses a few more problems than those we have discussed so far. The perhaps primary one relates to the fact that users do make mistakes or acquire the wrong information. We must expect erroneous values as inputs to the classifier. However, in an interactive system we have a chance to counter these errors as it is possible to ask the user to specify a doubtful attribute again. This reduces the risk that the diagnosis will be wrong or uncertain, and increases the fault-tolerance of the system.

Basically, what we need is a mechanism for detecting *inconsistencies* in the attribute values, which in this context means combinations of inputs which are very unlikely (but not necessarily impossible). In essence, we would like to check if any of the known input values are very unlikely given everything else we know about the situation. This can be directly formulated as calculating the likelihood that the conditional distribution of each attribute y given all other known attributes $\boldsymbol{x}$ generated the specific outcome,

$$\lambda_m^y = p(Y = y|\boldsymbol{x}, M) \tag{28}$$

where $\boldsymbol{x} = X_1 = x_1, \ldots, X_n = x_n$ are the known attributes and M the model parameters. In our case, the expression above can be written as

$$\lambda_m^y = \sum_{z \in Z} p(z|\boldsymbol{x}) \sum_{k \in P_z} \left(\pi_{z,k} \prod_{i=1}^{n} p_{z,k}(Y = y|Z = z) \right) \tag{29}$$

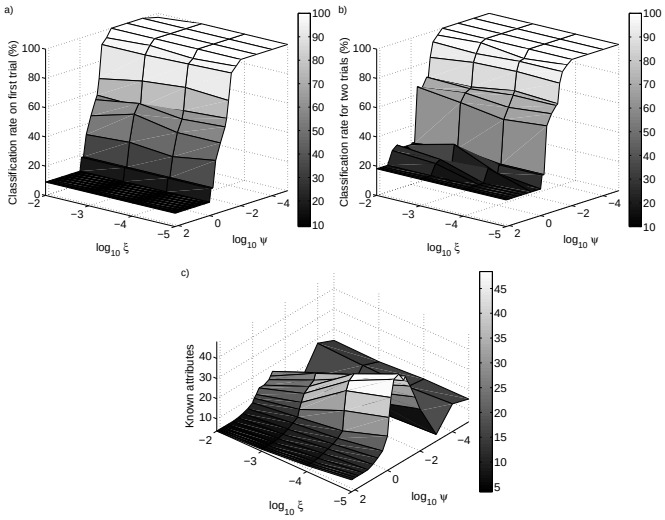

Fig. 6. Average diagnostic performance on the evaporation dataset for varying ξ and ψ, with 20% noise based on the prior distribution for each attribute within each class. The figure shows a) the classification rate on the first trial, b) the combined classification rate for two trials and c) the number of known attributes needed for a final hypothesis.

where $p(z|\boldsymbol{x})$ is calculated from equation 11. To find a suitable limit κ_m on λ_m^y for when to alert the user, we just need to define below what level of probability a value should be considered a possible inconsistency. The exact value certainly depends on the type of application and the nature of its domain, but we have consistently been using $\kappa_m = 0.05$ throughout our tests with good results.

A related problem occurs if we would like to be able not only to adapt the model to new cases, but, if the current input vector is inconsistent with the current model, suggest to the user that a new prototype should be created based on these inputs. Generally, we would like to decide whether or not the current input vector of known attributes should be regarded as normal or not given the current model. This can be done by calculating the likelihood λ_p that this input vector was generated by the model, or

$$\lambda_p = p(\boldsymbol{x}|M) \tag{30}$$

where $\boldsymbol{x} = X_1 = x_1, \ldots, X_n = x_n$ are the known attributes and M the model parameters, and can in our case be found directly from equation 11.

As we can see, calculating the likelihood of a pattern is straightforward, but below what level of likelihood should we start considering the pattern to be sufficiently abnormal? Again, let us start with the whole input vector. Assuming all prototypes should be interpreted as normal, a reasonable limit on the likelihood κ_p can be estimated as a fraction τ_p of the minimum of the likelihoods that each

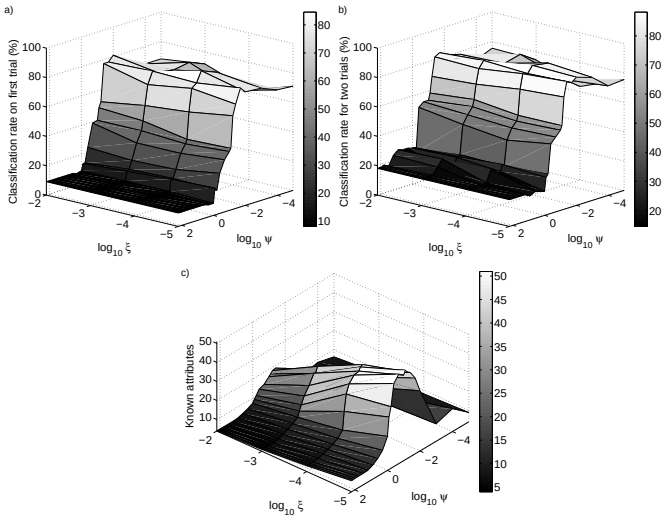

Fig. 7. Average diagnostic performance on the evaporation dataset for varying ξ and ψ, with 20% noise based on the prior distribution for each attribute over the whole dataset. The figure shows a) the classification rate on the first trial, b) the combined classification rate for two trials and c) the number of known attributes needed for a final hypothesis.

prototype was generated by the model,

$$\kappa_p = \tau_p \min_i p(\boldsymbol{x}_i|M) \tag{31}$$

where $\boldsymbol{x}_i$ are the attribute values of prototype i that are known in the current input pattern. A suitable value of τ_p then depends on how early we would like to trigger an inconsistency warning.

A problem that relates to the inconsistency checking described above is that of managing settings and prerequisites for a system. A system to be diagnosed often requires a number of prerequisites to be able to diagnose certain kind of class at all. A very simple example of this could be the requirement of having the ignition on in a car to be able to diagnose a fault in its electrical system. Most of the time, these requirements are much more complicated than this, and to enumerate all combinations of settings that constitute a possible user error is impossible simply because the vast majority of settings are incorrect. Only a few combinations are in fact valid settings. If this is the case, it is not practical to formulate this as a diagnosis problem, as we cannot easily find representative examples of all types of errors.

Instead, let us consider the possibility of specifying all combinations that represent *correct* settings. If this is indeed possible, as it often is in practical applications, we can estimate a statistical model based on data that representing different kinds of correct prerequisites. We can then detect *anomalies* in the

input vector by estimating the likelihood of a new pattern being generated by the model in the exact same manner as for detecting inconsistencies in expression 30.

Although used differently, the demands on the model used for this type of anomaly detection are very similar to those of the classification model we have used for diagnosis. We would like to combine prior system knowledge, which can be expressed as prototypical data, with actual usage cases in order to adapt the model to current circumstances. We will therefore here use exactly the same kind of model, the difference being that the classes will not represent a certain kind of *condition* to be diagnosed, but rather a certain kind of *scenario* for which the settings are valid. As before, one class or scenario can have many typical expressions, and thus many prototypes.

This allows us to calculate reasonable limits for when the likelihood of a certain pattern should be considered abnormal just like we did when we needed to detect inconsistencies through equation 31. If a pattern of settings is considered abnormal, we can naturally also use expression 28 to determine which settings attributes are most likely to be wrong.

Just like the diagnosis situation, we may not actually know all relevant information initially. We would therefore potentially like to perform the anomaly detection incrementally. However, calculating and presenting the expected reduction in entropy in the class distribution for each unknown attribute provides little information. We do not want to determine what class (condition) the pattern indicates, but rather to determine whether the pattern seems to belong to any of the classes at all. Therefore, we would like to rank our unknown attributes according to the *expected reduction in likelihood* to the whole pattern if we learn the value of one attribute Y,

$$G_L(Y) = p(\boldsymbol{x}|M) - E_Y[p(\boldsymbol{x}, Y|M)] \tag{32}$$

where $\boldsymbol{X}$ are the already known attributes, and E_Y the expectation according to Y. If Y is discrete, this can be written as

$$G_L(Y) = p(\boldsymbol{x}|M) - \sum_{y_j \in Y} p(Y = y_j|\boldsymbol{x}) p(\boldsymbol{x}, Y = y_j) \tag{33}$$

As before, there is no requirement that Y must be discrete, but the calculation of the expectation in equation 32 may be complicated if it is not. The expected reductions in likelihoods are presented to the user, who inputs new information accordingly.

This allows us to efficiently perform incremental anomaly detection, but there are certain practical limits to what we can detect. Without other attributes than those representing the settings or prerequisites, it is impossible for us to qualitatively separate two correct settings from each other. The specified settings may be acceptable, but unsuitable for the specific kind of diagnose we would like to perform. Introducing input attributes that in some way represent what kind of diagnosis the user wants to perform could provide a solution, as would the possibility of the user actually specifying the class attribute of the model,

representing the current diagnosis scenario. Note also that if this class is unknown, propagating its conditional distribution to the actual diagnosis model could improve diagnosis performance since it provides an idea of what the current scenario is. This of course depends on the availability of specified scenario attributes in the training data for the diagnosis model.

5.1 Experiments

We have investigated the diagnostic performance when using inconsistency checks on all the discrete datasets and on the continuous dataset. For this purpose, we performed two series of experiments in which the degree of noise was gradually increased.

In the first series of experiments, we measured the diagnostic performance without using inconsistency checks to obtain baseline performance. The baseline results were then compared to the diagnostic performance in the second series of experiments in which inconsistency checking was used. In both series of experiments the noise level was set to 20%, 35% and 50%. All of the experiments on both types of datasets were repeated 10 times in order to obtain statistical significance on the diagnostic performance.

Discrete datasets In the discrete case, we performed experiments using each dataset as prototypes in the model. Noise was induced by setting a random selection of attributes to the incorrect value based on the prior taken over the whole dataset, as described. The likelihood limit for all attributes in the discrete case was set to $\kappa_m = e^{-3}$. The prototype significance was set fixed to $\psi = 1.0$ while the thresholding parameter varied as $\xi = \{4.7 \times 10^{-6}, \ldots, 4.7 \times 10^{-1}\}$.

We observe that the diagnostic performance improves by the use of inconsistency checking (fig. 8-10). For example, the percentage of correctly diagnosed samples on the terrain-vehicle dataset increased by 9.03 percentage points for 50% noise compared to the corresponding baseline result when $\xi = 4.7e{-}6$ (tab. 5). In tab. 4-6, we observe that the number of anomaly checks in general matches the number of incorrect attribute values. In addition, our results indicate that the number of known attributes needed to obtain a final hypothesis (including the 'extra' inconsistency questions) can be reduced, compared to when not using inconsistency checking (fig. 8-10).

Continuous dataset In the continuous case, the samples in the dataset were used as cases in the model, whereas a fixed subset of samples was used to set attribute values, as earlier described. Noise was induced based on the Gaussian prior estimated from the sample mean and standard deviation of each attribute taken over the whole dataset.

Setting an inconsistency limit κ_m for an attribute is more complicated in the continuous case compared to the discrete case. For a discrete distribution, we can easily find the actual probability of a certain outcome. In the continuous case, we can find the probability density at a certain value. However, this is

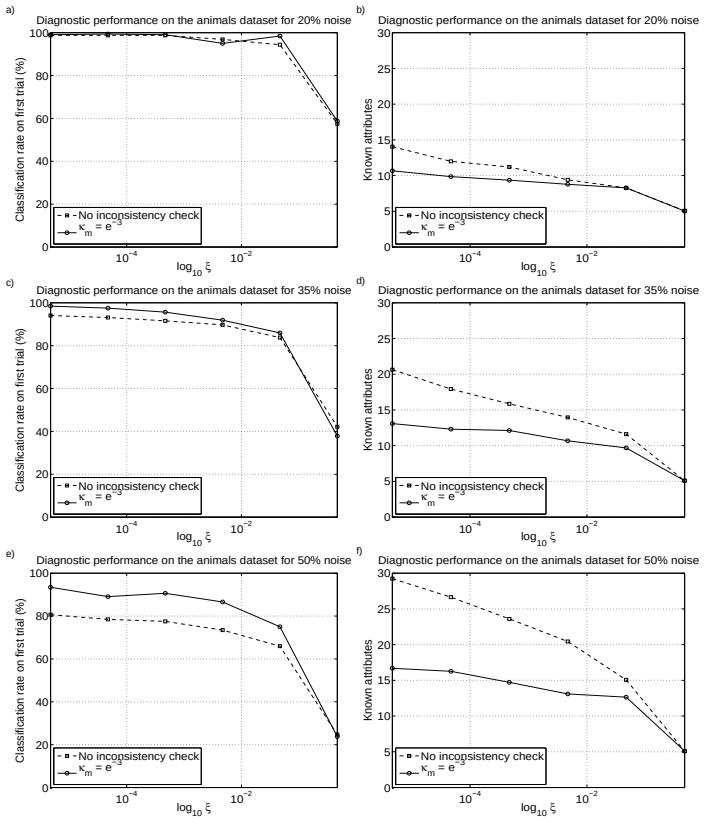

Fig. 8. Average diagnostic performance on the animals dataset using inconsistency checking for varying ψ and ξ: a,c,e) the total diagnostic performance from one trial: b,d,f) the number of known attributes needed for a final hypothesis.

not a proper probability, as it in a sense depends on the scale of the attribute, possibly assuming values larger than one. Here, we will therefore set the limit κ_m on the probability density individually for each continuous attribute to the probability density for which a certain fraction of the total probability mass is lower in the Gaussian prior estimated from the complete set of samples. This can be calculated from

$$\kappa_m = \frac{e^{-\mathrm{erf}^{-1}(c-1)^2}}{\sqrt{2\pi\sigma_m^2}} \tag{34}$$

where σ_m^2 is the estimated variance of attribute m, c the desired fraction, and erf^{-1} is the inverse error function. This will hopefully provide us with a rough approximation of an appropriate value for the limit, but not necessarily a very good one.

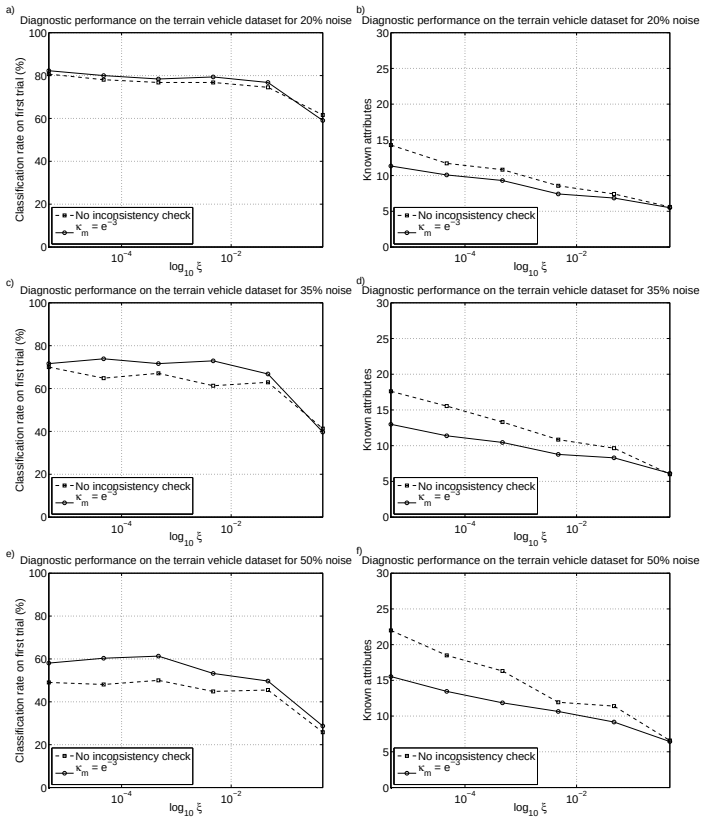

Fig. 9. Average diagnostic performance on the terrain vehicle dataset using inconsistency checking for varying ψ and ξ: a,c,e) the total diagnostic performance from one trial: b,d,f) the number of known attributes needed for a final hypothesis.

Ultimately, we would like to set the limit based on the conditional distribution of the attribute given the already known values, as the scale of this distribution might change drastically when further attributes become known. However, as this distribution is expressed as a mixture of Gaussians, calculating the proper value of the limit analytically becomes impossible. Therefore, we will settle for the approximation described above in our experiments.

In the experiments, the inconsistency limit κ_m was based on a fraction $c = e^{-3} \approx 0.05$ of the probability density function for each continuous attribute. For the discrete attributes the inconsistency limit was strictly set to e^{-3}. Thus, the inconsistency limit varied between different attributes. Here, the prototype significance was set to $\psi = 10^{-5}$ and the thresholding parameter to $\xi = 10^{-5}$.

We observe that the diagnostic performance on continuous data can be improved by the use of inconsistency checking. For example, we see that the diag-

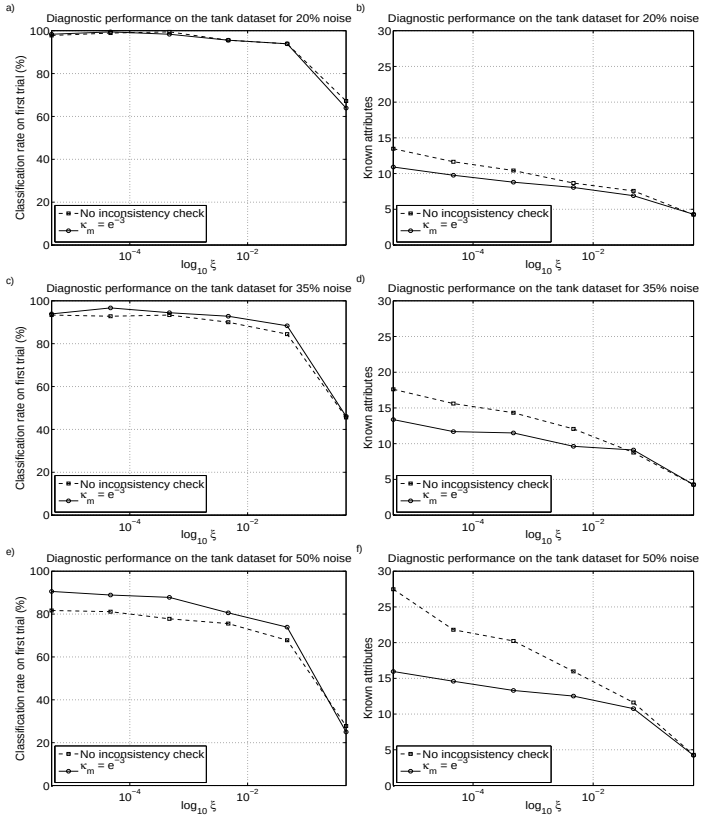

Fig. 10. Average diagnostic performance on the tank dataset using inconsistency checking for varying ψ and ξ: a,c,e) the total diagnostic performance from one trial: b,d,f) the number of known attributes needed for a final hypothesis.

nostic performance improves 13.64 percentage points when 20% noise is induced compared to the baseline experiments (tab. 7). However, the number of known attributes needed to obtain a final hypothesis does not seem to be significantly affected in general. We also observe that the number of anomaly checks does not fully match the number of incorrect attributes as well as in the discrete case. This is partly due to the fact that detecting inconsistencies in the continuous case is a more delicate matter than in the discrete case, and partly that the data continuous data set contain more ambiguities than the tested discrete data sets.

Since we have observed that the average percentage of correctly diagnosed samples may vary about 5% between different runs of the experiments, further repetitions of the experiments may be needed in order to obtain statistically reliable results. Observations during the experiments also indicate that the inconsistency limits κ_m may have to be re-calculated using another value on c

Without anomaly checking				
Noise (%)	Correct (%) Trial #1	Known attributes	Incorrect attributes	Anomaly checks
20	98.75	14.04	1.23	0
35	94.06	20.64	2.59	0
50	80.62	29.23	4.71	0
With anomaly checking				
Noise (%)	Correct (%) Trial #1	Known attributes	Incorrect attributes	Anomaly checks
20	99.06	10.67	0.91	1.31
35	98.44	13.09	1.78	2.49
50	93.44	16.69	3.16	4.13

Table 4. Average diagnostic performance on the animals dataset for $\psi = 1.0$ and $\xi = 4.7 \times 10^{-6}$ when using inconsistency checks.

Without anomaly checking				
Noise (%)	Correct (%) Trial #1	Known attributes	Incorrect attributes	Anomaly checks
20	80.65	14.27	1.09	0
35	70.00	17.61	2.13	0
50	49.03	21.98	3.45	0
With anomaly checking				
Noise (%)	Correct (%) Trial #1	Known attributes	Incorrect attributes	Anomaly checks
20	82.26	11.34	0.88	1.20
35	71.61	12.99	1.77	2.20
50	58.06	15.53	2.91	3.01

Table 5. Average diagnostic performance on the terrain vehicle dataset for $\psi = 1.0$ and $\xi = 4.7 \times 10^{-6}$ when using inconsistency checks.

(equation 34) when the degree of noise is changed. In spite of this, our results indicate that our approach of inconsistency checking could be used to obtain higher diagnostic performance on continuous data.

Without anomaly checking				
Noise (%)	Correct (%) Trial #1	Known attributes	Incorrect attributes	Anomaly checks
20	97.78	13.48	1.19	0
35	93.33	17.61	2.44	0
50	81.67	27.47	4.97	0
With anomaly checking				
Noise (%)	Correct (%) Trial #1	Known attributes	Incorrect attributes	Anomaly checks
20	98.33	10.92	0.97	1.16
35	93.89	13.37	2.07	2.45
50	90.56	15.96	2.97	3.32

Table 6. Average diagnostic performance on the tank dataset for $\psi = 1.0$ and $\xi = 4.7 \times 10^{-6}$ when using inconsistency checks.

Without anomaly checking				
Noise (%)	Correct (%) Trial #1	Known attributes	Incorrect attributes	Anomaly checks
20	65.45	10.39	2.15	0
35	49.09	8.41	2.95	0
50	36.36	7.36	3.69	0
With anomaly checking				
Noise (%)	Correct (%) Trial #1	Known attributes	Incorrect attributes	Anomaly checks
20	79.09	10.13	2.06	1.03
35	63.64	8.96	3.03	1.10
50	40.00	6.60	3.26	1.15

Table 7. Average diagnostic performance on the evaporation dataset for $\psi = 10^{-5}$ and $\xi = 10^{-5}$ when using inconsistency checks.

6 Corrective measures

After an incremental diagnosis session has been taken as far as practically possible, we often would like to propose a suitable corrective measure to the user, a way to manage the diagnosed problem. This can be rather simple, as one diagnose often corresponds directly to one corrective measure: For one diagnosed problem, there is only one solution. As long as there is a description of the correc-

tive measure associated with each possible diagnosis in the database, suggesting corrective measures is trivial.

If this is not the case, as e.g. in many situations within medical diagnosis, we can still provide the user with valuable information by presenting similar earlier case descriptions which can suggest a suitable treatment. What constitutes a similar case though is not necessarily obvious, but we can at least differentiate between a number of general alternatives.

Ultimately, we would like to find cases that are likely to have *similar corrective measures*. Unless the diagnosis corresponds directly to a corrective measure as discussed earlier, we will assume that we do not have sufficient information to determine this in the database. A good approximation of this, which is indeed often the correct one if a diagnosis are directly connected to a corrective measure, is to determine how similar two cases are by determining how *similar their class distributions* are given the known inputs. This can be done by calculating the *Kullback Leibler distance* [21] between the class distributions,

$$D(p_1||p_2^i) = \sum_{z \in Z} p_1(z) \log \frac{p_1(z)}{p_2^i(z)} \tag{35}$$

where $p_1(z)$ denotes the class distribution of the current diagnosis and $p_2(z)$ the class distribution of the i:th case, using the same known input variables as for the current diagnosis. Although the measure is highly useful for ranking cases by similarity to the current diagnosis, it is relatively safe to assume that distances closer to or above $\log |Z|$, where $|Z|$ denotes the number of outcomes in Z, show that the two distributions have very little similarity and can safely be assumed not to be related.

It is not unlikely that each of the prototypes of the database can be associated with a distinct corrective measure, even if the class itself cannot. In this situation, and indeed many others were knowing what prototype corresponds the best to the current diagnose situation can be beneficial to the user, we can for the model used easily calculate the probability distribution over the prototypes given the known inputs as

$$p(k|\boldsymbol{x}) = \sum_{z \in Z} p(Z = z) \pi_{z,k} \prod_{i=1}^{n} p_{z,k}(x_i|Z = z) \tag{36}$$

This allows us to present the user the prototypes the current inputs are most likely to be drawn from, along with the probability. It can only point to relevant prototypes while ignoring other data in the form of cases, but the computational complexity is significantly lower than calculating the distance in class distribution as above for each available case. If suitable, though, we can always calculate the distance to each case through an expression similar to equation 35, but using the prototype distribution instead of the class distribution.

Lastly, there is always an opportunity to calculate the similarity between cases by determining the *distance in input space* between two patterns. Here, however, our statistical model is of limited assistance when defining a suitable

distance measure, and we refer the reader to the case based reasoning literature instead.

7 Designing incremental diagnosis systems

Let us now review and summarise the methods discussed earlier by describing a more complete and practically useful incremental diagnosis system. In a wider setting, we would like to not only offer a diagnosis, but also suggest corrective action based on this diagnosis and collect information from new diagnostics cases. Fig. 11 shows a simplified view of the complete process.

First, *diagnosis* is performed incrementally until we have arrived at an acceptable hypothesis. This can be determined by the system as in the experiments of section 4.1, or by the user. The diagnostic procedure basically involves two parallel tasks: one to perform the actual diagnosis and one to determine whether the prerequisites for performing this diagnosis are fulfilled or not.

In the diagnostics case, the user first enters new information based on the presented entropy gains. The known input attributes are then checked for inconsistencies as described in section 5, and these inconsistencies signalled to the user. If relevant for the specific application, the prerequisites or settings are checked so that they are likely be correct. If not, information about what settings that are most probably incorrectly set should be shown to the user. If must be acquired at a cost, the user could be re-directed to perform this anomaly detection incrementally, as discussed in section 5.

After verifying the settings, a new classification is performed where the entropy gain is calculated for the unknown attributes and the conditional class distribution presented to the user. If we by now have arrived at an acceptable hypothesis, we can move on to assist the user in finding suitable *corrective measures*.

As we have described earlier, suggesting corrective measures is often rather straightforward as there is only one or a few suitable actions to take for a certain diagnose. However, if this is not the case, we can assist the user by finding the relevant prototypical and case data as in section 6.

Finally, we usually want the system to *adapt* to new cases. Before entering the case into the database and adapting the model to it, the case should be validated by the user so that the diagnosis specified is actually the correct one in order to avoid reinforcement of erroneous classifications by the model. If the actual diagnosis is new and not represented in earlier data, the case should instead be refined into a new prototype to give the system an opportunity to perform a correct classification in the future. A new prototype should potentially also be suggested if the case seem to represent a new representation of the class that is very dissimilar from earlier examples. This can be done by using the methods described in section 6.

The user interface to the diagnosis system should reflect the possibility of answering questions in any order or at any point changing the answer of an earlier question. This is a significant leap in flexibility compared to many expert

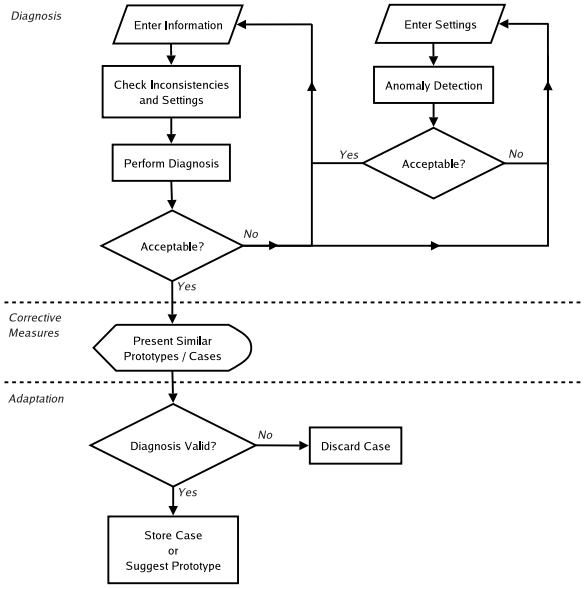

Fig. 11. A simplified view of the complete diagnostics process.

system approaches, and to design an interface that only allows the user to answer the currently most important question as in a traditional system defeats some of the advantages of the approach. Therefore, it is more beneficial to e.g. present a list of unknown attributes, ordered by their respective entropy gain, that can be set in arbitrary order along with earlier set attributes.

Fig. 12 shows an early prototype interface intended for use in the field by armed forces to diagnose technical equipment. It allows for selecting different types of technical equipment, and through that change the current database for the statistical model. The diagnosis can be focused on certain sections of the equipment, selected in a tree structure at the top of the application. The table to the left shows all possible questions and their current entropy is gain shown as a bar in the table. The top right table shows all possible diagnosis, sorted in descending order by their current probability which is also shown as a bar in the table. The frames below show instructions on how to repair the selected condition and what resources are necessary for the operation, and on the bottom of the window there is a button which adds the current case to the database.

This interface is likely to change, but can still serve as a useful example of what an interface to the diagnosis system may look like.

8 Discussion

We have shown a flexible, robust and efficient incremental diagnosis system trained from empirical data. Since the system can make use of both expert

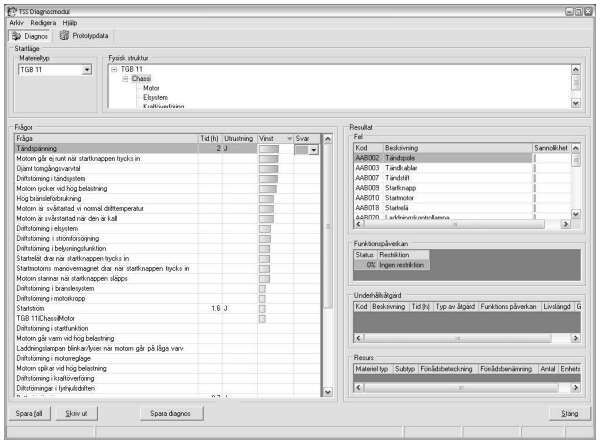

Fig. 12. A prototype interface for diagnosing technical materiel in the armed forces.

knowledge and examples of actual diagnosis cases, it provides a powerful alternative to rule based systems. Also, because the presented statistical model does not work with explicit rules, it does not have the same problem as rule based systems with inconsistencies in the data. Exceptions and ambiguities are handled in a natural way.

A highly desirable feature of the system for practical use is the fact that questions can be answered in arbitrary order. It is possible to avoid answering a certain question, answering another one instead, or even change a previous reply at any point. Initially providing some information also makes the diagnosis start from there, without having to traverse questions in a predefined order. This is highly usable if there e.g. are some automatic readings available or some system checks have already been performed.

Also, in spite of not using a rule based approach, we can still get simple explanations from the system, in terms of the primary causes for the systems' conclusion. Since it is difficult to build user confidence without providing explanations of the diagnosis systems conclusions, this property is critical in practical use. The fact that the system at every point provides a distribution over the classes also helps build user confidence and makes the diagnosis task easier.

Inconsistencies in the inputs can be reliably detected, giving the user an opportunity to correct faulty inputs to the system. The same mechanisms can also be used to determine whether the prerequisites to performing the classification are fulfilled or not. This also makes it possible to identify user errors, something that is indeed very common in practical situations.

Being somewhat similar to an instance based learner, the model does suffer from some of the same type of problems. Most importantly, as the number of prototypes increase in the system, so does the complexity of performing classification and by consequence the complexity of calculating the information gain

for unknown attributes. This effect is not necessarily present in other models such as just using one naive Bayesian classifier, for which the classification complexity would remain constant with the number of prototypes. However, adding new cases to the database, which in time most likely would constitute the bulk of the historical data, imposes no increase in computational complexity.

The fact that the system can both easily learn from each new diagnosed case and determine whether the case may be a suitable candidate for refinement into a prototype, are also very important in many settings. The properties of the diagnosed system is usually not completely known when it is taken into use, making the possibility of collecting and incorporating knew information continuously absolutely crucial. This way, the diagnosis system also provides a means for structuring and storing knowledge that might otherwise be lost with the departure of an experienced operator.

Future work includes testing the system on more practical cases to determining the performance of the model. These practical experiences would hopefully also allow us to deduct heuristics for adjusting the balance of significance between prototypical data and actual case data so that the system works well in all applications.

Acknowledgements

The authors would like to thank the Swedish procurement agency, FMV, for providing insight and understanding on how to perform and support diagnosis of technical systems. We would also like to thank Palmarium AB for their work on the user interfaces to the diagnosis system for the Swedish armed forces.

References

1. T. Cover. Estimation by The Nearest Neighbour rule. In *IEEE Transactions on Information Theory* 14(1) (1968) 50–55.
2. G. McLachlan and D. Peel. *Finite Mixture Models*. Wiley & Sons (2000).
3. I.J. Good. *Probability and the Weighting of Evidence*. Charles Griffin (1950).
4. M. Stensmo, A. Lansner and Björn Levin. A Query-Reply System based on a Recurrent Bayesian Artificial Neural Network. In *Proc. of the 1991 International Conference on Artificial Neural Networks, Espoo, Finland.* North-Holland, Amsterdam (1991).
5. M. Stensmo. *Adaptive Automated Diagnosis*. PhD Thesis, dept. of Computer and Systems Sciences, Royal Institute of Technology, Stockholm (1995).
6. A. Holst and A. Lansner. A Flexible and Fault Tolerant Query-Reply System Based on a Bayesian Neural Network. In *International Journal of Neural Systems* 4(3) (1993).
7. A. Veneris, J. Liu, M. Aimiri, M. Abadir. Inremental diagnosis and correction of multiple faults and errors. In *Design Automation and Test in Europe Conference and Exhibition (Date'02)* (2002).
8. R. Martinez-Bejar, F. Ibanez-Cruz, P. Compton. A reusable framework for incremental knowledge acquisistion. In *The fourth Australian Knowledge Acquisition Workshop*. University of South Wales, Sydney, 1999.

9. A. Wichert. Associative diagnosis. *Expert Systems* 22(1) (2005).
10. R. Schmidt, L. Gierl: The Roles of Prototypes in Medical Case-Based Reasoning Systems. In *5th German Workshop on Case-Based Reasoning, GWCBR-97*. Universität Kaiserslautern (1997).
11. PROMEDAS: a probabilistic decision support system for medical diagnosis. Technical report,Foundation for Neural Networks, Nijmegen, The Netherlands (2002).
12. H. J. Kappen. The Cluster Variation Method for approximate reasoning in medical diagnosis. In *Modeling Bio-medical signals*, World Scientific 2002.
13. H. J. Kappen, W. Wiegerinck, and E. ter Braak. Decision support for medical diagnosis. *The future of data mining*, STT (2001).
14. H. J. Kappen, W. Wiegerinck, E. ter Braak, W. ter Burg, M.J. Nijman, Y.L. O, and J.P. Neijt. Approximate inference for medical diagnosis. *Pattern Recognition Letters* (1999).
15. F. V. Jensen. *An introduction to Bayesian networks*. UCL Press (1996).
16. J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of plausible inference*. Morgan Kaufmann Publishers, Inc. (1988).
17. D. E. Heckerman, E. J. Horvitz, and B. N. Nathwani. Towards normative expert systems: part I, the Pathfinder project. *Methods of information in medicine*, 31:90–105, 1992.
18. D. E. Heckerman, and B. N. Nathwani. Towards normative expert systems: part II, probability-based representations for efficient knowledge acquisition and inference. *Methods of information in medicine*, 31:106–116, 1992.
19. R. A. Howard. Information value theory. *IEEE Transactions on Systems Science and Cybernetics*, 22–26, 1966.
20. C. E. Shannon. A mathematical theory of communication. *Bell Systems Technical Journal*, 27:379–423, 623–656, 1948.
21. S. Kullback, and R. A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22:79–86, 1951.
22. A. Holst. The use of a Bayesian neural network model for Classification Tasks. PhD thesis TRITA-NA-P9708, dept. of Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm, Sweden, 1997.