Biol. Cybern. 61, 361–370 (1989)

Biological Cybernetics
© Springer-Verlag 1989

Bayesian Neural Networks

I. Kononenko
Faculty of Electrical and Computer Engineering, Trzaska 25, YU-61000 Ljubljana, Yugoslavia

Abstract. A neural network that uses the basic Hebbian learning rule and the Bayesian combination function is defined. Analogously to Hopfield's neural network, the convergence for the Bayesian neural network that asynchronously updates its neurons' states is proved. The performance of the Bayesian neural network in four medical domains is compared with various classification methods. The Bayesian neural network uses more sophisticated combination function than Hopfield's neural network and uses more economically the available information. The "naive" Bayesian classifier typically outperforms the basic Bayesian neural network since iterations in network make too many mistakes. By restricting the number of iterations and increasing the number of fixed points the network performs better than the naive Bayesian classifier. The Bayesian neural network is designed to learn very quickly and incrementally.

1 Introduction

There is a variety of designs of neural networks. Neural networks can be classified according to various criteria, e.g. network topology, the aim of its usage, the learning rule and the combination function that combines the inputs of a single neuron into its output.

Topologically a network can be without layers (e.g. Hopfield and Tank 1985), with two layers (e.g. Minsky and Papert 1969; Kosko 1988), multilayered (e.g. Rumelhart et al. 1986a) and of some special topology (e.g. Kohonen 1984; Rumelhart and Zipser 1986). A network can be used as an auto-associative memory (e.g. Kohonen 1984; McEliece et al. 1987), as a hetero-associative memory (e.g. Kosko 1988), temporal associative memory (e.g. Kosko 1988), as a classifier (e.g. Rumelhart et al. 1986a), for clustering (e.g. Rumelhart and Zipser 1986) or for some special purpose (e.g. Hopfield and Tank 1985; Kohonen 1984). Most common learning rules are Hebbian and generalized Hebbian learning rule (Rumelhart et al. 1986), delta rule (Minsky and Papert 1969), generalized delta rule (Rumelhart et al. 1986a) and competitive learning (Rumelhart and Zipser 1986). A combination function for calculating neuron's output from its input can be deterministic or stochastic (e.g. Rumelhart et al. 1986). Most common combination functions include variants of weighted sum and sigma-pi functions (Rumelhart et al. 1986b).

This paper is primarily concerned with the *combination function*, while using specific *topology* and *learning rule*. The same techniques described in this paper can be used also for other topologies. The application of these techniques is not fixed, however, for demonstration purposes a neural network was designed for classification. It is shown that a neural network can naturally implement "*naive*" *Bayesian classifier*. In next section the Bayesian neural network is defined and the convergence for a network that asynchronously updates its states is proven. In Sect. 3 experimental results in four medical domains are described. The performance of the Bayesian network is improved by increasing the number of fixed points. The results are compared with some other classification methods and with the performance of human experts. Finally, in Sect. 4 the results are commented and some conclusions are given.

2 Definition of Bayesian Neural Network

2.1 Topology and Learning Rule

The most general topology (without layers) is chosen where every neuron in a network is connected with every other neuron via bidirectional connections, called synapses. In addition to the memory cell that stores the weight for each synapse two additional

memory cells (counters) for each neuron are introduced whose function will be explained later. A state of a neuron corresponds to its output. The neuron states are restricted to two possible values: 1 (active) and 0 (inactive). Two different topologies will be used: one in which each neuron is connected to itself via a *feedback connection* and one in which those connections are omitted.

A network works in two distinct phases: *learning* and *classification.* During the learning phase the input patterns (learning instances) are presented to the network and weights and counters of the network are modified. During the classification the memory of the network is fixed. Every neuron receives an *input* and produces an *output* in the following way. Every neuron receives an initial state from an input pattern by adopting the corresponding component value of the vector. The neurons then asynchronously and iteratively change their states according to the *combination function.* The network terminates with the computation when no more neurons can change their states, i.e. when the *fixed point* is reached. When the neural network finishes with computation the component values of the output pattern are extracted from the corresponding neurons' states.

At the beginning of learning all weights are set to 0. The learning rule used is the basic Hebbian rule (Rumelhart et al. 1986b) which states that the weight of a connection between two neurons is increased (in our case by 1) if both neurons are active. Therefore the weight of a given synapse is proportional to the frequency with which the two components of the learning patterns (that correspond to two neurons connected via the given synapse) are both 1.

At the same time, learning takes place also in two additional memory cells of each neuron. One of the cells is increased each time an input pattern (a learning instance) is presented to the network. Therefore, this cell in fact represents a counter of learning instances. The second memory cell in each neuron is increased whenever a neuron is active. Therefore, its value is proportional to the frequency with which the corresponding component in learning instances equals 1.

Before proceeding with the definition of a design of the Bayesian neural network a "naive" Bayesian classifier is developed, as used also by Bratko and Kononenko (1987).

2.2 "Naive" Bayesian Classifier

Let the objects from a given domain be described with a fixed number of *attributes* (features, descriptors). Each attribute has a fixed number of possible values. Each object from a given domain is defined with a set of attributes' values, one value for each attribute. Let V_{ij} be a Boolean variable which is true (has value 1) if a given object has j-th value of i-th attribute and false (has value 0) otherwise. If the value of i-th attribute is unknown then for given i all V_{ij} are 0. Note that for each attribute i at most one V_{ij} has value 1.

Let S be a conjunction of conditions $V_{ij}=1$ for all V_{ij} that have value 1 for a given pattern to be classified. Furthermore, for brevity, $P(V_{ij})$ will be used instead of $P(V_{ij}=1)$ for prior probability that V_{ij} has value 1. If the value for m-th attribute is not given we can compute for each value n of that attribute its probability by computing the probability that $V_{mn}=1$ with the following Bayesian formula:

$$P(V_{mn}|S)=P(V_{mn})P(S|V_{mn})/P(S),$$

where $P(X)$ is prior probability of X being true and $P(X|Y)$ is the probability that X is true given Y. If we assume that attributes are independent (because of this assumption a classifier is called "naive") we get

$$P(S)=\prod_{V_{ij}=1} P(V_{ij}),$$

$$P(S|V_{mn})=\prod_{V_{ij}=1} P(V_{ij}|V_{mn}).$$

Since

$$P(V_{ij}|V_{mn})=P(V_{ij}\ \&\ V_{mn})/P(V_{mn})$$

we get

$$P(V_{mn}|S) = P(V_{mn}) \prod_{\substack{V_{ij}=1 \\ mn\neq ij}} [P(V_{ij}\ \&\ V_{mn})/(P(V_{ij})P(V_{mn}))]. \qquad (1)$$

The independence assumption is needed to obtain (1) which can be effectively computed by approximating the probabilities on the right hand side with relative frequencies obtained from the set of learning patterns.

2.3 Combination Function

We now continue with the definition of a design of a Bayesian neural network from Sect. 2.1. We assume that each learning instance (input vector) represents an object in a given domain. An object is described with values of a fixed set of attributes. One attribute with n possible values is represented with n neurons (n vector components). Therefore, for each value V_{ij}, as defined in Sect. 2.2, we have one neuron. If an object has j-th value for i-th attribute then the neuron is set to 1, otherwise it is set to 0. If the value for i-th attribute is not known then for given i all V_{ij} are set to 0.

If we assume that prior probabilities can be approximated with relative frequencies from a set of learning instances then the weight of a synapse connecting two neurons which represent values V_{mn} and V_{ij} corresponds to $P(V_{mn}\ \&\ V_{ij})$ times the number of

learning examples. The two counters in a neuron representing value V_{mn} correspond to the number of learning instances and $P(V_{mn})$ times the number of learning instances, respectively.

It remains to define a *combination function* with which a single neuron combines its input into output (which also represents its state). The *activation level* of a neuron representing V_{mn} given a state of a network S is

$$A(V_{mn}|S)=P(V_{mn}|S),$$

where $P(V_{mn}|S)$ is calculated from (1). Note that if (1) is used on real data (where attributes need not be independent) then the value greater than 1 can be obtained. Activation level is not a probability but is proportional to the probability. The combination function for a neuron representing value V_{mn} is

$$V'_{mn}=\begin{cases} 1 & \text{if } A(V_{mn}|S)>P(V_{mn}) \\ V_{mn} & \text{if } A(V_{mn}|S)=P(V_{mn}) \\ 0 & \text{if } A(V_{mn}|S)<P(V_{mn}), \end{cases} \tag{2}$$

where V'_{mn} represents the new neuron's state. According to this combination function, the new state is equal 1 if the activation level is greater than prior probability and 0 if it is less than prior probability. If both values are the same then the neuron's state remains unchanged.

The above discussion is valid for the topology with no feedback connections. For the topology where each neuron is connected to itself we generalize (1) by omitting the condition $mn \neq ij$:

$$A(V_{mn}|S) = P(V_{mn}) \prod_{V_{ij}=1} [P(V_{ij} \,\&\, V_{mn})/(P(V_{ij})P(V_{mn}))]. \tag{3}$$

We used the notation A instead of P because (3) mathematically does not represent the probability. If V_{mn} is given to be 1 then $P(V_{mn}|S)=1$. Anyway, (3) is the natural generalization of (1) if a neuron makes no distinction among the connections to other neurons and the connection to itself. If V_{mk} is given to be 1 then for $n=k$ we get from (3):

$$A(V_{mn}|S)=\prod_{\substack{V_{ij}=1 \\ mn\neq ij}} [P(V_{ij} \,\&\, V_{mn})/(P(V_{ij})P(V_{mn}))]$$

and for all $n \neq k$ we get:

$$A(V_{mn}|S)=0$$

since $P(V_{mn} \,\&\, V_{mk})$ equals 0 for $n \neq k$ and equals $P(V_{mn})$ if $n=k$. Furthermore, if the value of m-th attribute is unknown then (1) and (3) are identical.

Note that all information necessary to compute V'_{mn} is *locally* available to each neuron. Each neuron stores also the number of all learning instances which is needed for approximation of probabilities with relative frequencies.

When a network finishes with learning the weights are fixed and the network can be used to process new, unseen objects, for which some values of attributes are missing or are wrong. All neurons will work in parallel by iteratively changing their states according to states of other neurons. Missing values will be approximated and the wrong values will be possibly corrected. The network finishes its work when there are no more changes of neuron states. In the next section we prove that networks with both kinds of topology, defined in Sect. 2.1, will terminate their work in a finite time regardless of input vector and current state of weights in a network if neurons work *asynchronously*.

Note that in the beginning only one V_{ij} for given attribute i may be active (equal 1) at the same time and all the others must be 0. The other possibility is that all V_{ij} for given i are 0 (the value of i-th attribute is unknown). When the neurons operate asynchronously the only possible change is that a single V_{ij} changes from 1 to 0 or, in the latter case, that a single V_{ij} changes from 0 to 1. If, on the other hand, the neurons operate synchronously more than one V_{ij} for given i may change from 0 to 1. But in the next iteration all V_{ij} will be set to 0 since they will inhibit each other.

2.4 *Convergence Proof*

The proof of the convergence is analogous to those of Hopfield (McEliece et al. 1987) and Kosko (1988) for networks which use the symmetric matrix obtained as the sum of outer products of learning vectors as their memory. Hopfield and Kosko define a function which Hopfield calls the *energy function*. They show that the energy monotonously decreases if the network *asynchronously* updates its neurons' states. Since the function has a lower bound the process must terminate in a finite time. Hopfield's energy function (Kosko 1988) is given by inner product

$$E=-1/2XMX^T,$$

where X is a vector describing a current state of a network and M is a *symmetric* matrix obtained as the sum of outer products of learning vectors and represents the network's memory. Note that here the states of neurons can be 1 or -1. The product MX^T is a vector which in fact represents *activation levels* of neurons given a current state of a network. The function therefore measures the dissimilarity between the neurons state and activation levels. The higher is the activation level for a neuron with state equal 1 (the lower is the activation level for a neuron with state equal 0) the greater is *similarity* and, analogously, the higher is the activation level for a neuron with state

equal 0 (the lower for a neuron with state equal 1) the greater is dissimilarity.

Kosko (1988) defines a similar function for bidirectional associative memories:

$$E = -BM^T A^T = -AMB^T,$$

where M is the memory matrix obtained as the sum of outer products among pairs of learning vectors and (A, B) is a pair of vectors describing the current state of a network. The vector $M^T A^T$ represents in fact the activation levels of neurons in layer "B" given a current state of layer "A". Similarly holds for vector MB^T for neurons in layer "A". Therefore if we omit the minus sign we again obtain the *measure of similarity.*

We now define a measure of similarity between a current state of a Bayesian neural network and activation levels of neurons given a current state:

$$\mathrm{Sim}(S) = \prod_{V_{ij}=1} [A(V_{ij}|S)/P(V_{ij})],$$

where S again represents a current state of a network. Obviously, this function has an upper bound since it has a finite number of possible values if the prior probabilities are fixed.

Note that if $A(V_{ij}|S) > P(V_{ij})$ and $V_{ij}=1$ then the similarity is in fact greater. The new value of V_{ij} in that case will tend to stay 1, and vice versa. Values V_{ij} that have current value 0 are ignored by the similarity function. If the value of attribute i is known, e.g. $V_{ik}=1$, then all V_{ij} are 0 for $j \neq k$ and new values V'_{ij} will tend to stay 0. If, on the other hand, the value of attribute i is unknown (all V_{ij} are 0) then attribute i is currently ignored by the similarity measure.

Now we prove that if only one neuron changes its state at a time (network works *asynchronously*) then the similarity will increase. In the previous section it was already stated that because of asynchronicity at most one V_{ij} for given i is equal 1 at the same time. First we prove this for the topology where the connection of a neuron to itself is forbidden. Therefore, $A(V_{ij}|S)$ is defined with (1).

As only one neuron is allowed to change at a time it suffices to consider one change alone. The change can be of two types:

a) Old $V_{mn}=1$ and $A(V_{mn}|S) < P(V_{mn})$ and new V_{mn} is $V'_{mn}=0$:

Let S' describe a new state after the change. Then we have

$$\mathrm{Sim}(S')/\mathrm{Sim}(S)$$
$$= P(V_{mn})/A(V_{mn}|S) \prod_{V_{ij}=1} [A(V_{ij}|S')/A(V_{ij}|S)]$$
$$= P(V_{mn})/A(V_{mn}|S) \prod_{\substack{V_{ij}=1 \\ ij \neq mn}} [P(V_{ij})P(V_{mn})/P(V_{ij} \,\&\, V_{mn})]$$
$$= [P(V_{mn})/A(V_{mn}|S)]^2 > 1.$$

b) Old $V_{mn}=0$ and for all j: $V_{mj}=0$ and $A(V_{mn}|S) > P(V_{mn})$ and new V_{mn} is $V'_{mn}=1$:

$$\mathrm{Sim}(S')/\mathrm{Sim}(S)$$
$$= A(V_{mn}|S')/P(V_{mn}) \prod_{\substack{V_{ij}=1 \\ ij \neq mn}} [A(V_{ij}|S')/A(V_{ij}|S)]$$
$$= A(V_{mn}|S')/P(V_{mn}) \prod_{\substack{V_{ij}=1 \\ ij \neq mn}} [P(V_{ij} \,\&\, V_{mn})/(P(V_{ij})P(V_{mn}))]$$

and since $A(V_{mn}|S') = A(V_{mn}|S)$ we get

$$= [A(V_{mn}|S)/P(V_{mn})]^2 > 1.$$

The proof for the topology where each neuron is connected to itself is analogous to the above proof and only results are given. In this case $A(V_{ij}|S)$ is defined with (3). We have the same two types of changes. For type a) we obtain:

$$\mathrm{Sim}(S')/\mathrm{Sim}(S) = P(V_{mn})/A(V_{mn}|S)^2 > 1,$$

and for type b) we have:

$$\mathrm{Sim}(S')/\mathrm{Sim}(S) = A(V_{mn}|S)^2/P(V_{mn})^3 > 1.$$

The results also indicate the faster convergence for (3).

3 Experiments in Four Medical Domains

To test the performance of the Bayesian neural network we used it as a classifier in four medical domains. In all domains the problem is similar: given the description of a patient in terms of attributes' values the system must return one of possible diagnoses. For each domain we have a fixed set of attributes and a fixed number of possible diagnoses. A diagnosis is in our case treated as an additional attribute with values corresponding to separate diagnoses.

Table 1 shows the characteristics of four medical domains. The number of neurons is the sum of values of all attributes plus the number of diagnoses. The percentage of cases belonging to majority class in the domain is also added in the table. We obtained the data from The University Clinical Center in Ljubljana. Data include descriptions of a certain number of patients with known diagnoses.

3.1 Experiments with Basic Bayesian Neural Network

One run was performed by randomly selecting 70% of instances for learning and 30% for testing. Learning instances were used to train the network by appropriately computing synapses' weights and neurons' counters. When learning was completed the network's memory was fixed. The network was then tested by running it *asynchronously* (in every iteration only one neuron was allowed to change its state) on every testing instance with corresponding diagnosis deleted (all the

Table 1. Characteristics of four medical domains

Domain name	#attrib.	#class.	#cases	#neur.	maj. class
Primary tumor	17	22	339	59	25%
Breast cancer	10	2	288	29	80%
Thyroid diseases	15	4	884	141	56%
Rheumatism	32	6	355	298	66%

neurons representing the diagnosis were set to 0). When the network converged to a fixed point the answer was obtained by seeking for an active neuron among neurons standing for diagnoses. If none of them was active we took as an answer diagnosis j that maximized $A(V_{ij}|S)$, where S is a final state of the network and i is the attribute representing the diagnosis.

In each experiment a run was repeated 10 times and an average was taken. The following parameters were measured:

- *classification accuracy:* percentage of correctly classified testing instances
- *information content per answer:* the average amount of information contained in one system's answer (Kononenko and Bratko 1989). This measure avoids the problems with classification accuracy when one class is much more probable then the others. In Table 1 one can see that in all four medical domains this is indeed the case. The definition of the information content of an answer is provided in the Appendix.
- *average number of iterations:* average number of changes of neurons' states necessary to reach a fixed point.

We measured also the average number of changes of states from 0 to 1. In all experiments this number was approximately 1/3 to 1/2 of the average number of iterations.

Two groups of experiments were performed in each medical domain. First, for the neural network topology where connections of each neuron to itself are omitted (1), and the other, for the topology with feedback connections (3). In each group we performed three different experiments with respect to the criterion for selecting the neuron that can change its state. Recall that only one neuron is allowed to change its state at a time although there may be more neurons that satisfy the criterion for changing the state (i.e. $V'_{mn} \neq V_{mn}$ where V'_{mn} is obtained from (2)).

In the first experiment a neuron was selected *randomly*. In the second experiment a neuron was also selected randomly but with the *probability* proportional to its calculated $A(V_{ij}|S)$, for those who should change from 0 to 1, and proportional to $1 - A(V_{ij}|S)$ for those who were to change their states from 1 to 0. In the third and last experiment, the neuron which *maximized* $A(V_{ij}|S)$ was selected if the change was from 0 to 1, and $1 - A(V_{ij}|S)$ if the change was from 1 to 0. Note that the first two criteria need only *locally* available information and that the last criterion needs *global* information and therefore cannot be naturally implemented in a neural network.

Table 2. Results of experiments in the "primary tumor" domain

Equation	Criter.	Accuracy	Inf. cont.	#iterat.
(1)	Rand.	28.7%	0.74 bits	16.2
(1)	Prob.	36.2%	1.10 bits	13.9
(1)	Max.	37.2%	1.15 bits	11.9
(3)	Rand.	31.5%	0.90 bits	9.8
(3)	Prob.	39.5%	1.19 bits	9.4
(3)	Max.	41.0%	1.27 bits	8.7

Overall there were six experiments (with ten runs) each for every medical domain. Results for the primary tumor domain are presented in Table 2. For other domains the percentages are absolutely different but relatively have same trends for different experiments as in the primary tumor domain. In all medical domains the topology where each neuron is connected to itself performed better and required a smaller number of iterations. The topology with feedback connections tries to preserve the current state of the network while the topology without feedback connections tries to change it. We will explain why the former strategy is better in the next section.

As expected, the random selection of a neuron that was allowed to change its state was the worst with respect to classification accuracy and the number of necessary iterations. The selection of the neuron with the maximal difference between the calculated and the prior probability outperformed the other two criteria, both with respect to classification accuracy, and the number of iterations.

In Table 3 the performance of the Bayesian neural network with feedback connections and the selection criterion that maximizes the difference between the calculated and the prior probability is compared to the performance of the naive Bayesian classifier. Note that the naive Bayesian classifier is equivalent to the Bayesian neural network with zero iterations.

Table 3. Comparison of performances of the Bayesian neural network and the naive Bayesian classifier in four medical domains

Domain	Bayesian neural network			"Naive" Bayes	
	Accuracy	Inf. cont.	#iterat.	Accuracy	Inf. cont.
Primary tumor	41.0%	1.27 bits	8.7	47.1%	1.42 bits
Breast cancer	72.5%	0.04 bits	3.1	77.6%	0.15 bits
Thyroid	61.9%	0.63 bits	7.2	67.8%	0.73 bits
Rheumatism	47.3%	0.18 bits	21.7	50.5%	0.21 bits

3.2 Increasing the Memory Capacity of the Neural Network

The Bayesian neural network performed worse than the "naive" Bayesian classifier. The independence assumption affected more the performance of the network since the network works *multidirectionally* and the "naive" Bayesian classifier works only in one direction (from attributes to classes). The assumption that attributes are independent is a smaller mistake than the same assumption in parallel with the assumption for each attribute that other attributes (including an attribute representing classes) are independent.

It seems that iterations in the neural network, while trying to replace missing data and correct wrong data, are making too many mistakes. When the equilibrium is reached the original pattern is too corrupted so that the calculated probabilities of diagnoses are also affected.

The problem can be stated also in terms of the *memory capacity* of the Bayesian neural network, i.e. the number of different patterns that can be stored by the network. For classification purposes, the ideal network would have the capacity equal to the number of different patterns from a given domain. For example in the primary tumor diagnostic problem described with 17 attributes (each with 2–4 possible different values) there are approximately 4.4 millions of different descriptions of patients assuming that each description corresponds to exactly one diagnosis (which is not always the case).

In this paper the momory capacity of the Bayesian neural network is not studied in detail. Obviously the capacity is finite and far below the desired. There have been various studies of the capacity of various types of neural networks (Guez et al. 1988; McEliece et al. 1987; Wong 1988). The usual way to increase the memory capacity of the neural network is the development of special learning rules. However, the memory capacity is strongly related to the number of *fixed points* obtained by learning. By increasing the number of fixed points we can expect that also the memory capacity will increase.

An obvious way to increase the number of fixed points is to strengthen the condition for changing the neuron's state. In one extreme the condition is never satisfied and all the input patterns become fixed points. It was already stated that such Bayesian neural network without any iterations, used as a classifier, is equivalent to the naive Bayesian classifier. However, iterations in the neural network can be useful to approximate missing data and correct wrong data.

The other extreme is the combination function defined with (2) which suffices that the network will converge into a fixed point in a finite time. This function was used in our experiments so far. In the next experiment the following combination function was used instead of (2):

$$V'_{mn}=\begin{cases} 1 & \text{if } A(V_{mn}|S)\geqq 1 \\ V_{mn} & \text{if } 1>A(V_{mn}|S)>0 \\ 0 & \text{if } A(V_{mn}|S)=0. \end{cases} \quad (4)$$

Note again that $A(V_{mn}|S)$ can also be greater than 1 because of the independence assumption used to calculate it. This combination function strengthens the conditions from (2), therefore the convergence is guaranteed. However, the conditions are very strong and it can be expected that the number of iterations necessary to reach a fixed point will significantly decrease (the convergence will be faster). The neurons will change their states only if there will be a lot of evidence for the change. The results given in Table 4 confirm the above discussion and show that the Bayesian neural network using the combination function defined with (4) performs better than the naive Bayesian classifier.

It is interesting that with the introduction of the combination function (4) *all types of Bayesian neural networks perform almost identically.* There is no difference between the performance of both topologies and, as well, the performance is independent to the criterion for selecting the neuron to change its state (as almost in all iterations only one neuron satisfied the condition to change its state).

The introduction of combination function (4) significantly increased the number of fixed points. Some of them correspond to training instances and the others cover the problem space not covered by the

Table 4. Performance of the Bayesian neural network using combination function (4) compared to the performance of the naive Bayesian classifier

Domain	Bayesian neural network			"Naive" Bayes	
	Accuracy	Inf. cont.	#iterat.	Accuracy	Inf. cont.
Primary tumor	47.9%	1.46 bits	0.6	47.1%	1.42 bits
Breast cancer	78.1%	0.15 bits	0.4	77.6%	0.15 bits
Thyroid	68.2%	0.74 bits	1.7	67.8%	0.73 bits
Rheumatism	59.7%	0.42 bits	10.3	50.5%	0.21 bits

Table 5. Performance of the Bayesian neural network using combination functions (2) and (4) on training instances

Domain	Combination function (2)			Combination function (4)		
	Acc.	Inf. cont.	#iterat.	Acc.	Inf. cont.	#iterat.
Primary tumor	55.5%	1.91 bits	8.3	60.5%	2.06 bits	0.6
Breast cancer	72.7%	0.11 bits	2.8	79.7%	0.19 bits	0.4
Thyroid	71.9%	0.89 bits	5.8	75.2%	0.95 bits	0.6
Rheumatism	82.0%	1.32 bits	8.6	84.8%	1.38 bits	3.9

Table 6. The comparison of performance for different classifiers in the "primary tumor" domain. (A) indicates the implementation of the naive Bayesian classifier in the Assistant learning system

Classifier	Primary tumor		Breast cancer		Thyroid		Rheumatism	
	Acc.	Inf. cont.	Acc.	Inf. cont.	Acc.	Inf. cont.	Acc.	Inf. cont.
Bayesian n.n.	48%	1.46 bits	78%	0.15 bits	68%	0.74 bits	60%	0.42 bits
Naive Bayes	47%	1.42 bits	78%	0.15 bits	68%	0.73 bits	50%	0.21 bits
Naive Bayes (A)	49%	1.59 bits	79%	0.06 bits	68%	0.70 bits	57%	0.28 bits
Assistant	44%	1.38 bits	77%	0.07 bits	73%	0.87 bits	61%	0.46 bits
Specialists	42%	1.22 bits	64%	0.05 bits	64%	0.59 bits	56%	0.26 bits
Nonspecial.	32%	0.95 bits	64%	0.03 bits	–	–	–	–

training instances. Not all the *training* instances are stored as can be seen from Table 5. In fact, domains used in our experiments are *incomplete* in the sense that with a given set of attributes the classification can not be exact as attributes carry too few information. Therefore the sensible generalization will not exactly classify all the training instances.

3.3 Comparison with Other Classification Methods

The above results are compared in Table 6 with:

(a) Naive Bayesian classifier as described in Sect. 2.2. It was already stated that the naive Bayesian classifier is equivalent to the Bayesian neural network with zero iterations (probabilities of classes are calculated without any change in neurons' states). Note also that (3) in that case is equivalent to (1) because the class is not given in advance. In Table 6 the results of the naive Bayesian classifier implemented in the Assistant learning program (Bratko and Kononenko 1987) are added [marked with (*A*)]. This implementation includes some modifications with respect to the reliability of the approximation of probabilities with relative frequencies.

(b) Hopfield's neural network that uses generalized Hebbian learning rule which states that the connection between two neurons is increased if their states are identical. The memory is therefore the sum of outer products of training vectors. States of neurons can be -1 (inactive) and 1 (active). Unknown value for an attribute in that case corresponds to 0 states of all corresponding neurons. The combination function is the usual weighted sum:

$$Y_i = f\left(\sum_j W_{ij} X_j\right),$$

where

$$f(X)=\begin{cases} 1 & \text{if } X>0 \\ \text{previous neuron's state} & \text{if } X=0 \\ -1 & \text{if } X<0. \end{cases}$$

We tested the same two topologies as for the Bayesian neural network. The topology where connections of a neuron to itself are omitted uses the memory matrix with 0 diagonal and the other topology uses matrix with all diagonal elements equal to the number of learning vectors. We used only the third criterion for the selection of a neuron which is allowed to change its state, the one with maximal absolute weighted sum.

In all our experiments in first three medical domains the Hopfield's neural network converged to a fixed point with the majority class as an answer. In the "rheumatism" domain approximately one half of testing instances were classified into the majority class and one half in the class with relative frequency 0.08.

We tested also Hopfield's neural network without any iterations (as an analogue to "naive" Bayesian classifier). In that case the weighted sum is calculated for each class from the initial state of a network and the returned class is the one with the maximal sum. Except for 3% of testing instances in the primary tumor domain which were classified into different classes this method performed in all experiments in all domains identically as Hopfield's neural network with iterations.

Thus in all domains Hopfield's neural network achieved much worse results than the Bayesian neural network. This could be expected since the coding of one pattern was designed for the Bayesian neural network. The performance of Hopfield's neural network can be improved by using a different encoding. Although such comparison is not appropriate, we encoded attributes' values using binary numbers (one neuron corresponding to one digit of the binary number). This encoding decreased the number of neurons in the network. Results were better but still much worse than that of the Bayesian neural network. Results of Hopfield's neural network (with and without iterations) are omitted from tables as the comparison is not appropriate.

(c) Inductive learning system Assistant (Bratko and Kononenko 1987; Cestnik et al. 1987) for constructing binary decision trees. In a decision tree nodes correspond to attributes, branches to values of attributes and leaves to classes. The following is the learning algorithm:

1. Select the most informative attribute and cluster its values into two subsets.

2. Split the training instances according to values of that attribute into two subsets.

3. For each subset of training instances do if stopping condition is satisfied then generate a leaf else recursively generate a subtree.

Assistant copes also with incomplete and noisy data. Decision trees are comprehensible to human experts while the decisions of Bayesian classifier cannot be directly interpreted. The attempts to interpret Bayesian classifier's decisions are described in (Kononenko 1989; Michie 1989). On the other hand learning of decision trees is much slower than learning the relative frequencies for Bayesian classifier.

(d) Human experts: physician specialists (in the case of primary tumor and breast cancer also nonspecialists) were tested on a set of randomly selected patients. The results presented in Table 6 are the averages of 4 physicians in each domain.

The experiments with classifiers (a), (b), and (c) were done in the same manner as with the Bayesian neural network. Ten runs were performed for each experiment and averages were calculated. For each experiment a subset of 70% of instances was randomly selected for training while the rest of the instances was used for testing.

4 Conclusions

The Bayesian neural network uses the available information more economically than Hopfield's neural network. Hopfield's neural network uses memory elements which are proportional to probabilities of two neurons being in the same state:

$$P(X_i=X_j).$$

Bayesian neural network uses probabilities that both neurons are active and probabilities that one is active:

$$P(X_i \,\&\, X_j), P(X_i), P(X_j)$$

which are more informative. Namely, the former probability can be derived from the latter ones, while the opposite is not the case:

$$\begin{aligned} P(X_i=X_j) &= P(X_i \,\&\, X_j)+P(\overline{X_i} \,\&\, \overline{X_j}) \\ &= P(X_i \,\&\, X_j)+[1-P(X_i)-P(X_j) \\ &\quad +P(X_i \,\&\, X_j)]. \end{aligned}$$

The combination function used by Hopfield's neural network could also be modified in order to increase the number of fixed points as was done for the Bayesian neural network in the previous section. However, there are no obvious lower and upper bounds as for the Bayesian neural network (namely 0 and 1). Anyway, Hopfield's neural network without iterations as an analogue to the naive Bayesian classifier performed

too badly that any significant improvement could be expected.

Assistant and the naive Bayesian classifier have similar performance although they use different information. Bayesian classifier takes into account all attributes but assumes their independence. Assistant uses only attributes that appear in a decision tree (discards useful information) but takes into account the dependencies between attributes that appear in a decision tree. Future work should concentrate on the learning algorithms for overcoming the independence assumption which usually limits the neural networks computation to *linearly separable* functions (Williams 1986).

The generalization effect in the Assistant learning scheme is the generation of *general* rules that cover a great part of the problem space. The same effect was in the Bayesian neural network with combination function (2). There were few fixed points covering great parts of the problem space. Because of the great number of iterations necessary to reach a fixed point the network can easily make a mistake and terminate in wrong fixed point. With the introduction of combination function (4) the generalization effect is the generation of many fixed points *specialized* for one or few instances from the problem space. Neighbor fixed points are similar and the transition from one fixed point to another is less dangerous with respect to the final decision.

The performances of Assistant, "naive" Bayesian classifier and Bayesian neural network with strengthened changing condition are comparable to the performance of human experts. This does not mean that physicians can be replaced by computers. When diagnosing a patient a physician takes into account also other sources of information which cannot be used by a computer. However, the comparison with human experts shows that learning systems almost optimally use the available information. Thus the systems can be used as efficient tools for improving and verifying the diagnostic process. In (Kononenko 1989) it is shown how a naive Bayesian classifier and the Bayesian neural network can "explain" their decisions.

It is interesting to see how noise in an input pattern affects the performance of the network. Figure 1 compares the performance of Bayesian neural network and the naive Bayesian classifier on noisy input patterns. Experiments in the "primary tumor" domain were repeated by corrupting the testing instances. To a certain percent of randomly selected attributes random values were assigned. The results are again averages of ten runs. Here the combination function defined with (4) was used. Only results for typology with connections of neurons to themselves and with the selection of the best neuron for changing its state

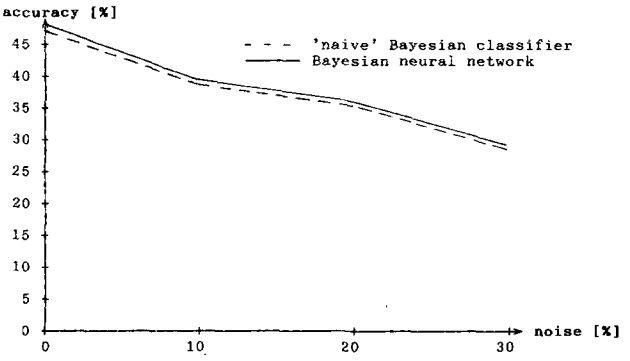

Fig. 1. Comparison of classification accuracy among the "naive" Bayesian classifier and the Bayesian neural network when testing instances are noisy in the "primary tumor" domain

are given. The results for the other topology and other selection criteria are almost identical.

The accuracy and the information content of answers for both systems decreased with the increased amount of noise in testing instances. All the time the accuracy of the network was slightly better than that of the naive Bayesian classifier. The similar result was obtained for information content. As expected, the number of necessary iterations increases with noise (from 0.6 with zero noise to 1.0 iterations with 30% of noise).

The Bayesian neural network is designed to work *multidirectionally*, i.e. it makes no distinction between input and output. Therefore it can be used also as a multidirectional classifier and as an associative memory as well. It learns very quickly (in our simulations typically a few seconds and not more than 30 s on an IBM PC-AT) and *incrementally*.

Acknowledgements. Dr. Sergej Hojker, dr. Vlado Pirnat and dr. Matjaz Zwitter from University Clinical Center in Ljubljana provided the medical data and tested the physicians' performance. I would like to thank Andrej Dobnikar, Sasa Dzuroski, and Franci Solina for their comments on the manuscript. I thank also the anonymous reviewer for constructive comments and suggestions.

Appendix: Definition of the Information Content of an Answer

A fair evaluation criterion has to exclude the influence of the prior probabilities of classes which may enable a completely uninformed classifier to trivially achieve high classification accuracy. The measure of information content of the classifier's answer defined below excludes the influence of prior probabilities, deals with various types of imperfect or probabilistic answers and can be used also for comparing the performance in different domains. Its interpretation is natural.

Definition. Let the correct class of an instance be C, $P(C)$ be prior probability of class C and $P'(C)$ probability of class C returned by a classifier. The *information content* I of classifier's answer is

defined as follows (note that in our experiments only *exact classification* is considered, i.e. $P'(C)$ is always either 1 or 0):

a) If $P'(C) > P(C)$ then

$$I = -\log_2 P(C) + \log_2 P'(C) \quad \text{[bits]}$$

i.e., the amount of obtained information is the entire amount of information necessary to correctly classify an instance into class C minus the remainder of information necessary to correctly classify that instance.

b) If $P'(C) = P(C)$ then $I = 0$ [bits]

i.e., the system didn't change the prior probability of the correct class therefore we didn't obtain any information.

c) If $P'(C) < P(C)$ then

$$I = -(-\log_2(1 - P(C)) + \log_2(1 - P'(C))) \quad \text{[bits]}$$

i.e., the amount of information returned by the system is the entire amount of information necessary to decide that an instance doesn't belong to class C minus the remainder of information necessary to make that decision. As this information is in fact wrong the information content of the system's answer in this case is negative.

References

Bratko I, Kononenko I (1987) Learning diagnostic rules from incomplete and noisy data. In: Phelps B (ed) Interactions in artificial intelligence and statistical methods. Technical Press, Hampshire

Cestnik B, Kononenko I, Bratko I (1987) Assistant 86: a knowledge elicitation tool for sophisticated users. In: Bratko I, Lavrac N (eds) Progress in machine learning. Sigma Press, Wilmslow

Guez A, Protopopsecu V, Barhen J (1988) On the stability, storage capacity and design of nonlinear continuous neural networks. IEEE Trans SMC-18:80–87

Hopfield JJ, Tank DW (1985) "Neural" computation of decisions in optimization problems. Biol Cybern 52:141–152

Kohonen T (1984) Self-organization and associative memory. Springer, Berlin Heidelberg New York

Kononenko I (1989) Interpretation of neural networks decisions. Proceedings of IASTED International Conference on Expert Systems, Zurich, Switzerland, June 26–28

Kononenko I, Bratko I (1989) Informativity based evaluation criterion for classifier's performance. Mach Learn J (to appear)

Kosko B (1988) Bidirectional associative memories. IEEE Trans SMC-18:49–50

McEliece RJ, Posner EC, Rodemich ER, Venkatesh SS (1987) The capacity of the Hopfield associative memory. IEEE Trans IT-33:461–482

Michie D (1989) Personal models of rationality. J Statist Planning and Inference (in press)

Minsky M, Papert S (1969) Perceptrons. MIT Press, Cambridge

Rumelhart DE, Zipser D (1986) Feature discovery by competitive learning. In: Rumelhart DE, McClelland JL (eds) Parallel distributed processing, vol 1: Foundations. MIT Press, Cambridge

Rumelhart DE, Hinton GE, Williams RJ (1986a) Learning internal representations by error propagation. In: Rumelhart DE, McClelland JL (eds) Parallel distributed processing, vol 1: Foundations. MIT Press, Cambridge

Rumelhart DE, Hinton GE, McClelland JL (1986b) A general framework for parallel distributed processing. In: Rumelhart DE, McClelland JL (eds) Parallel distributed processing, vol 1: Foundations. MIT Press, Cambridge

Williams RJ (1986) The logic of activation functions. In: Rumelhart DE, McClelland JL (eds) Parallel distributed processing, vol 1: Foundations. MIT Press, Cambridge

Wong AJW (1988) Recognition of general patterns using neural networks. Biol Cybern 58:361–372

Received: November 21, 1988

Accepted in revised form: May 4, 1989

Igor Kononenko, M.Sc.
Faculty of Electrical and Computer Engineering
Trzaska 25
YU-61000 Ljubljana
Yugoslavia