The Handling of Don't Care Attributes

Hahn-Ming Lee and Ching-Chi Hsu*

Department of Computer Science and Information Engineering
National Taiwan University, Taipei, Taiwan
Tel: 886-2-3917406, Fax: 886-2-3628167
E-mail: cchsu%twntucc1.bitnet@cunyvm.cuny.edu

ABSTRACT

A critical factor that affects the performance of neural network training algorithms and generalization of trained networks is the training instances. The troublesome problem occurs in specifying training instances in which some attributes are irrelevant to the decision. These irrelevant attributes are always set to zero indicating don't care. Although it is possible to train a neural network model to recognize the training instances specified in this way, the final network might be inconsistent with original information implied by these generalized instances with don't care attributes. The primary concern of this study is to handle the don't care attributes in training instances. Several approaches are discussed and their experimental results are also demonstrated.

* Please send all correspondences to this author.

CH 3065-0/91/0000-1085 $1.00 © IEEE

1. Introduction

In the past few years, neural networks have gain great attention. Researchers think it is helpful to solve machine intelligent problems and computationally intensive problems through the collective behavior of a large number of simple processing units. The training algorithm has been assumed to be an important feature to achieve the properties of intelligent information processing. A critical factor that affects the performance of training algorithms and generalization of trained networks is the training set [8]. Therefore, how to specify training instances is important for many applications. The troublesome problem occurs in specifying training instances in which some attributes are irrelevant to the decision. These irrelevant attributes are always set to zero indicating don't care [1,2]. Although it is possible to train a neural network to recognize the training instances specified in this way, the final network might not realize the information implied by don't care attributes.

For clarity, we use an example describe in [1] to illustrate the problem. That is, a node with a weight vector W and a threshold T is used to recognize two instances (1,1,D) and (-1,1,1), where D indicates don't care. If W * I - T > 0, then the input instance I is the desired instance. In this way, the possible weight vector W and threshold T can be W=(0,1,1) and T=0 if D is encoded as 0. Although this node with W=(0,1,1) and T=0 can recognize instances (1,1,0) and (-1,1,1), it cannot recognize (1,1,-1). Therefore, other approaches to handle don't care attributes must be discussed.

2. The Sample Problem and the Training Algorithm

The sample problem, Knowledge Base Evaluator 1 (KBE1), was used as an example to detail each approach that handles don't care attributes. The KBE1 is an expert system to evaluate expert system applications. It is a modification of the Knowledge Base Evaluator [3]. The output of KBE1 is Suitability, which can take the value: POOR (0) or GOOD (1), indicating that the application of the expert system on a domain is poor or good, respectively. The value of Suitability is determined by the attributes: worth, employee acceptance, solution available, easier solution, teachability, and risk. The possible attributes values and corresponding encoded values used are shown below: (The attributes values can be arbitrarily coded into numerical values)

worth: {HIGH, MODERATE, LOW, NEGATIVE} encoded as {2, 1, -1, -2},
employee acceptance: {POSITIVE, NEUTRAL, NEGATIVE} encoded as {2, 1, -1},
solution available: {ADEQUATE, PARTIAL, NONE} encoded as {2, 1, -1},
easier solution: {COMPLETE, PARTIAL, NONE} encoded as {2, 1, -1},
teachability: {FREQUENT, POSSIBLE, DIFFICULT} encoded as {2, 1, -1},
risk: {HIGH, MODERATE, LOW} encoded as {2, 1, -1}.

There are eighteen instances for Suitability [3] shown in Table 1. In the second and the third instances, the don't care attributes can be any values. These generalized instances must be transformed well so that the translated training instances are consistent with original ones. Since both of the generalized instances in Table 1 are negative instances, some other instances sets must be generated in order to investigate the behavior of each approach that handles don't care attributes. Thus, instances set in Table 1 is termed IS_1, and two other instances sets, termed IS_2 and IS_3, are generated. IS_2 is generated from IS_1 by (1) removing the two negative generalized instances; (2) adding in two positive generalized instances; and (3) deleting conflict instances and redundant ones. It is shown in Table 2. On the other hand, IS_3 is set up by adding to IS_2 the two negative generalized instances in IS_1. In other words, IS_1 only consists of negative generalized instances, IS_2 is generated to own positive generalized instances only, and IS_3 is set up to have both of the negative generalized instances and the positive ones.

In order to test whether the trained network based on the transformed instances is consistent with original instances or not, test instances sets, termed $T\text{-}SET_1$, $T\text{-}SET_2$, and $T\text{-}SET_3$, are formed by generating all the possible values for each don't care attribute. For instance, the second instance in IS_1 is translated into 3^5 training instances, and the third instance in IS_1 is transformed into 3^4 training instances. Therefore, $T\text{-}SET_1$ contains 340 training instances translated from the original 18 instances in IS_1. Then each instance in $T\text{-}SET_1$ is tested on the trained network and the number of misclassified instances, termed test-set error, is counted. Similarly, $T\text{-}SET_2$ and $T\text{-}SET_3$ are generated and tested in the same way. As a result, $T\text{-}SET_2$ contains 332 instances, and $T\text{-}SET_3$ consists of 656 instances.

Before proceeding into the methods that handle don't care attributes, we would like to describe the training algorithm used to characterize the behaviors of the transformed instances. The back-propagation algorithm [7], termed **bp**, based on a two-layer perceptron [5] was used to recognize transformed instances. The network contains one output node, and the number of hidden nodes was set as attributes number d * 2. Since there are some parameters that will affect the performance of **bp**, we tried the better one that has previously been used [6]. The learning rate is set to be 0.5 in an attempt to make it converge more quickly. Training terminates when the network correctly classifies all the transformed training instances, and the entire simulation is implemented on SUN-3/60.

3. Approaches to Handle Don't Care Attributes

In this section we describe several approaches to handle don't care attributes, and then discuss their problems.

Table 1. The instances set IS_1 for KBE1

instance number	attributes						Suitability
	worth	employee acceptance	solution available	easier solution	teachability	risk	
1	1	1	2	2	-1	2	0
2	-2	D	D	D	D	D	0
3	-1	D	D	D	D	2	0
4	2	-1	1	2	1	-1	1
5	-1	-1	-1	1	2	-1	0
6	2	-1	1	-1	-1	1	1
7	2	2	1	2	2	2	0
8	2	2	1	1	1	-1	1
9	-1	2	2	-1	2	-1	1
10	2	-1	1	-1	2	2	1
11	-1	2	-1	2	-1	1	0
12	-1	1	2	2	2	-1	1
13	1	2	2	-1	-1	2	0
14	2	-1	2	1	2	2	0
15	1	-1	-1	1	-1	2	1
16	2	2	-1	-1	2	-1	1
17	-1	1	-1	-1	-1	-1	1
18	1	1	2	1	-1	-1	0

* D: don't care

Table 2. The instances set IS_2 for KBE1

instance number	attributes						Suitability
	worth	employee acceptance	solution available	easier solution	teachability	risk	
1	1	1	2	2	-1	2	0
2	-1	-1	-1	1	2	-1	0
3	-1	2	2	-1	2	-1	1
4	-1	2	-1	2	-1	1	0
5	-1	1	2	2	2	-1	1
6	-1	1	-1	-1	-1	-1	1
7	1	2	2	-1	-1	2	0
8	1	-1	-1	1	-1	2	1
9	2	D	D	D	D	D	1
10	1	D	D	D	D	-1	1

* D: don't care

Approach 1: Replace don't care attributes with a fixed value

The first transformation method is to encode a new value for don't care attributes. For example, we encode don't care as 0. This approach was used in many applications [1,2]. In this way, each generalized instance is transformed into one training instance. The **bp** algorithm is then used to train the transformed instances. Next, the test sets T-SET_1, T-SET_2, and T-SET_3 are recognized by their corresponding trained networks. For example, IS_1 is transformed using this approach and trained by **bp**, then T-SET_1 is recognized by the trained network and the number of misclassified instances is 3. Similarly, IS_2 and IS_3 are both processed in the same way, and the numbers of test-set error are 84 and 88, respectively. The result is shown in Figure 1. A conclusion derived from this result is that although the trained networks can recognize the translated instances well, they are inconsistent with the information implied by don't care attributes. For instance, if the input instance (worth=NEGATIVE, employee acceptance=POSITIVE, solution available=NONE, easier solution=NONE, teachability= FREQUENT, risk=LOW) is recognized by the trained network based on the translated instances of IS_1, the response value for Suitability is GOOD. This violates the second instance in IS_1 since the attribute, i.e. worth=NEGATIVE, is dominated by other attributes.

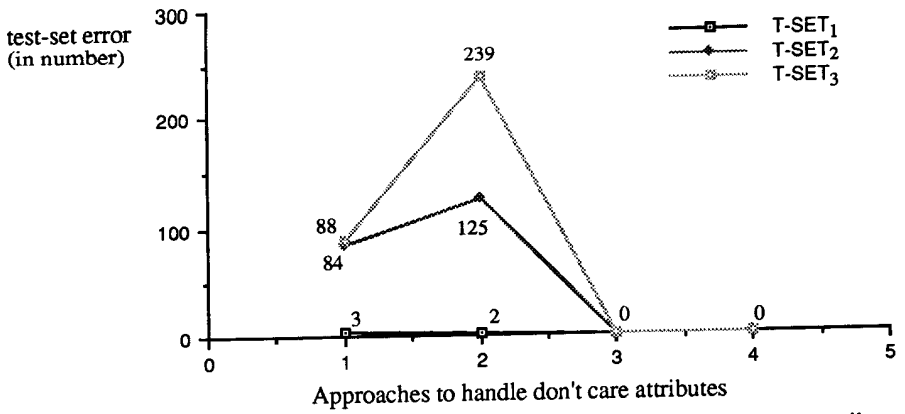

Figure 1. The test-set errors of various approaches that handle don't care attributes

In our experiments, it was found that this approach is not suitable, especially when positive generalized instances are encoded. It seems important to specify typical positive instances completely when **bp** is used. The above remark leads us to derive the following summary. That is, although this approach was used in many applications, the information implied by don't care attributes cannot be realized well.

Approach 2: Replace don't care attributes with their maximum or minimum encoded values

Before proceeding further, we first define several variables. MAX_i indicates the maximum encoded value of ith attribute, MIN_i means the minimum encoded value of ith attribute, DECIDE denotes the output of the system, and F_i is the value of ith attribute. Another proposed translation method is to replace don't care attribute F_i with MAX_i or MIN_i depending on the relationship between F_i and DECIDE. If F_i excites DECIDE, then the don't care attribute F_i is replaced with MIN_i; otherwise F_i is set to be MAX_i. The strategy used here is to encode these don't care attributes in "worst" situation, i.e. try to let specified attributes dominate don't care attributes in generalized instances. This idea comes from the failure of Approach 1. In this way, the attributes of the second instance in IS_1 are transformed into (-2,2,-1,-1,2,-1) and those of the third instance in IS_1 are translated into (-1,2,-1,-1,2,2). Other instances in IS_1 are unchanged. These transformed instances are then trained and test set $T\text{-}SET_1$ is recognized by the trained network. The number of this test-set error is 2. Similarly, IS_2 and IS_3 are processed in the same way, and the numbers of corresponding test-set error are 125 and 239. The experimental result is also shown in Figure 1. By this, we have to admit that this approach still cannot work well, especially when positive generalized instances are encoded. Besides, the conclusion we can give from the result is that **bp** does not directly make use of the relationship between attributes and DECIDE.

Approach 3: Replace don't care attributes with their maximum and minimum encoded values

Since instances transformed via the methods stated above are inconsistent with original information implied by don't care attributes, another approach was tried. Another strategy used is to replace don't care attribute F_i with MAX_i and MIN_i. Unlike one-to-one mapping as Approaches 1 and 2 stated above, this approach transforms one generalized instance into 2^k training instances, where k is the number of don't care attributes in this generalized instance. Using this approach, the original 18 instances in IS_1 are transformed into 64 training instances. Similarly, IS_2 is translated into 56 instances, and IS_3 is transformed into 104 instances. Also, the experimental result is shown in Figure 1. This result shows that **bp** can train, using this approach to handle don't care attributes, a neural network which is consistent with original instances.

Approach 4: Replace don't care attributes with all their possible encoded values

It is without saying that the trained network is consistent with original instances when the generalized instances are transformed using this approach. That is, the trained network will learn to ignore these don't care attributes and the transformed instances sets are $T\text{-}SET_1$, $T\text{-}SET_2$, and $T\text{-}SET_3$. This, however, expands the training set. If there are many possible values for a particular attribute, the training set expands even more.

Based on the discussion thus far, a conclusion can be drawn; that is, although replacing don't care attributes with their maximum and minimum values will expand instances sets, it is a feasible way to deal with don't care attributes. It can reduce a lot of transformed instances and training time , as compared with Approach 4, especially when there are many possible values for a particular attribute. The result is shown in Figure 2.

4. Discussion and Conclusion

The handling of don't care attributes is of primary concern in this paper. Although transforming don't care attributes into a fixed value was used in many cases and the trained network can recognize all of the transformed instances, it must be admitted that this trained network might be inconsistent with original information implied by don't care attributes. Replacing don't care attribute F_i with MAX_i and MIN_i seems to be a feasible solution when **bp** is used to distinguish these transformed instances. It has been demonstrated that this approach can work well when **bp** is used to train networks. However, not every algorithm can train, using Approach 3 to handle don't care attributes, a network which is consistent with original instances. For example, if the algorithm **bu** [4], described later, is used to train networks using the instances transformed by Approach 3, the numbers of test-set error for $T\text{-}SET_1$, $T\text{-}SET_2$, and $T\text{-}SET_3$ are 2, 1, and 1, respectively. Of course, if Approach 4 is used, any training algorithms can be utilized to

distinguish the transformed instances well. The instances set, however, expands even more.

Fortunately, **bu** can work well even for a large training set. It is a neural network training algorithm, in which the network topology does not have to be specified before training, to handle the classification problem with multi-valued inputs and binary output. The network topology is automatically generated in a finite number of time by incremental or non-incremental training with instances which define any arbitrarily complex decision regions. The operation of this algorithm can be written down as follows. The training process first selects one positive instance associated with all negative instances to form a convex region which covers this positive instance and none of the negative instances. Then another positive instance, which is not covered by the generated convex regions, is chosen to construct another convex region in the same way. The above process is repeated until all of the positive instances are covered by the generated convex regions. Consequently, the union of these constructed convex regions can be used to represent the training instances set.

Since **bu** can separate training instances into several modules and then train them individually, it is a good way to recognize the expanded training set. Even though the don't care attributes are replaced with all their possible values, **bu** can soon train a network to recognize all the transformed instances. The encouraging result is demonstrated in Figure 2.

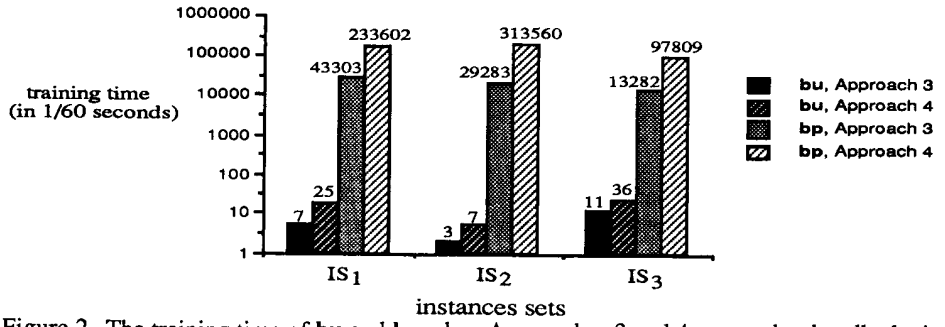

Figure 2. The training time of **bu** and **bp** when Approaches 3 and 4 are used to handle don't care attributes in instances sets IS_1, IS_2, and IS_3.

References

[1] H.Drucker," Implementation of Minimum Error Expert System," *IJCNN-90-San Diego*, vol. 3, 1990, pp 137-142.

[2] S.I.Gallant," Connectionist Expert Systems," *Comm. of the ACM*, vol. 31, no. 2, pp 152-169, Feb. 1988.

[3] R.Keller, *Expert System - Development & Approach*, Prentice-Hall, Inc., 1987.

[4] H.M.Lee and C.C.Hsu," Training of a Neural Network with Topology Generation for the Classification Problem," *Proceedings of International Neural Network Conference*, THOMSON-CSF/INNS/IEEE, Paris, July 1990.

[5] R.P.Lippmann," An Introduction to Computing with Neural Nets," *IEEE ASSP magazine*, pp 4-22, April 1987.

[6] R.Mooney, J.Shavlik, G.Towell, and A.Gove," An Empirical Comparison of Symbolic and Connectionist Learning Algorithms," *IJCAI 89*, 1989, pp 775-780.

[7] D.E.Rumelhart, G.E.Hinton, and R.J.Williams," Learning Internal Representations by Error Propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol 1: Foundations*, edited by D.E.Rumelhart and J.L.McClelland, The MIT Press, Cambridge, MA., 1986.

[8] M.Wann, T.Hediger, and N.N.Greenbaun," The Influence of Training Sets on Generalization in Feed-Forward Neural Networks," *IJCNN-90-San Diego*, 1990, vol. 3, pp 137-142.