90-30-4

A Comparison of ID3 and Backpropagation for English Text-to-Speech Mapping

Thomas G. Dietterich
Hermann Hild
Ghulum Bakiri
Department of Computer Science
Oregon State University
Corvallis, OR, 97331-3202

A Comparison of ID3 and Backpropagation for English Text-to-Speech Mapping

Thomas G. Dietterich
tgd@cs.orst.edu

Hermann Hild
s_hild@irav1.ira.uka.de

Ghulum Bakiri
haya@cs.orst.edu

Department of Computer Science
Oregon State University
Corvallis, OR 97331-3202

Abstract

The performance of the error backpropagation (BP) and ID3 learning algorithms was compared on the task of mapping English text to phonemes and stresses. Under the distributed output code developed by Sejnowski and Rosenberg, it is shown that BP consistently outperforms ID3 on this task by several percentage points. Three hypotheses explaining this difference were explored: (a) ID3 is overfitting the training data, (b) BP is able to share hidden units across several output units and hence can learn the output units better, and (c) BP captures statistical information that ID3 does not. We conclude that only hypothesis (c) is correct. By augmenting ID3 with a simple statistical learning procedure, the performance of BP can be approached but not matched. More complex statistical procedures can improve the performance of both BP and ID3 substantially. A study of the residual errors suggests that there is still substantial room for improvement in learning methods for text-to-speech mapping.

1 Introduction

The task of mapping English text into speech is quite difficult. A complete text-to-speech system involves many stages of processing. Ideally, sentences are parsed to identify word senses and parts of speech. Individual words (and their senses) are then mapped into strings of phonemes and stresses. The phonemes and stresses can then be combined by various techniques to generate sound waves. For an excellent review, see Klatt (1987).

In this paper, we focus on the task of mapping isolated words into strings of phonemes and stresses as formulated by Sejnowski and Rosenberg (1987) in their widely known work on NETTALK. A phoneme is an equivalence class of basic sounds. An example is the phoneme /p/. Individual occurrences of a /p/ are slightly different, but they are all considered /p/ sounds. For example, the two p's in "lollypop" are pronounced differently, but they are both members of the equivalence class of phoneme /p/. We use 54 phonemes (see Appendix A).

Stress is the perceived weight given to a syllable in a word. For example, the first syllable of "lollypop" receives the primary stress, the third syllable receives secondary stress, and the middle syllable is unstressed. Stress information is coded by assigning one of six possible stress symbols to each letter. Consonants generally receive one of the symbols "<" or ">", which indicate that the principal vowel in this syllable is to the left or the right (respectively) of the consonant. Vowels are generally marked with a code of 0 (none), 1 (primary), or 2 (secondary) to indicate the degree of stress. Lastly, silent stress ("–") is assigned to blanks.

Let L be the set of 29 symbols comprising the letters **a–z**, and the comma, space, and period (in our data sets, comma and period do not appear). Let P be the set of 54 English phonemes and S be the set of 6 stresses employed by Sejnowki and Rosenberg. The task is to learn the mapping f:

$$f : L^* \longrightarrow P^* \times S^*.$$

Specifically, f maps from a word of length k to a string of phonemes of length k and a string of stresses of length k. For example,

```
f("lollypop") = ("lal-ipap", ">1<>0>2<").
```

Notice that letters, phonemes, and stresses have all been aligned so that silent letters are mapped to the silent phoneme /–/.

As defined, f is a very complex discrete mapping with a very large range. If we assume no word contains more than 28 letters (the length of "antidisestablishmentarianism"), this range would contain more than 10^{70} elements. Existing learning algorithms focus primarily on learning Boolean concepts—that is, functions whose range is the set $\{0, 1\}$. Such algorithms cannot be applied directly to learn f.

Fortunately, Sejnowski and Rosenberg developed a technique for converting this complex learning problem into the task of learning a collection of Boolean concepts. They begin by reformulating f to be a mapping g from a seven-letter window to a single phoneme and a single stress. For example, the word "lollypop" would be converted into 8 separate 7-letter windows:

```
g("___loll") = ("l", ">")
```

```
g("__lolly") = ("a", "1")
g("_lollyp") = ("l", "<")
g("lollypo") = ("-", ">")
g("ollypop") = ("i", "0")
g("llypop_") = ("p", ">")
g("lypop__") = ("a", "2")
g("ypop___") = ("p", "<")
```

The function g is applied to each of these 8 windows, and then the results are concatenated to obtain the phoneme and stress strings. This mapping function g now has a range of 324 possible phoneme/stress pairs, which is a substantial improvement.

Finally, Sejnowski and Rosenberg code each possible phoneme/stress pair as a 26-bit string, 21 bits for the phoneme and 5 bits for the stress. Each bit in the code corresponds to some property of the phoneme or stress. This converts g into 26 separate Boolean functions, $h_1, \ldots, h_{26}$. Each function h_i maps from a seven-letter window to the set $\{0, 1\}$. To assign a phoneme and stress to a window, all 26 functions are evaluated to produce a 26-bit string. This string is then mapped to the nearest of the 324 bit strings representing legal phoneme/stress pairs. We used the Hamming distance between two strings to measure distance. (Sejnowski and Rosenberg used the angle between two strings to measure distance, but they report that the Euclidean distance metric gave similar results. In tests with the Euclidean metric, we have obtained results identical to those reported in this paper.)

With this reformulation, it is now possible to apply Boolean concept learning methods to learn the h_i. However, the individual h_i must be learned extremely well in order to obtain good performance at the level of entire words. This is because errors aggregate. For example, if each h_i is learned so well that it is 99% correct and if the errors among the h_i are independent, then the 26-bit string will be correct only 77% of the time. Because the average word has about 7 letters, whole words will be correct only 16% of the time.

In the remainder of this paper, we describe a series of experiments comparing the performance of the error backpropagation algorithm (BP) to the decision-tree learning algorithm ID3. We begin by comparing BP and ID3 on the task described above. Having established that BP significantly outperforms ID3 on this task, we formulate three hypotheses to explain this difference. We test these hypotheses by performing additional experiments. These experiments demonstrate that ID3, combined with some simple statistical learning procedures, can nearly match the performance of BP. Finally, we present data suggesting that there is still substantial room for improvement of learning algorithms for text-to-speech mapping.

2 A Simple Comparative Study

In this study, ID3 and BP were both applied to the learning task described above. We begin by briefly reviewing these two learning algorithms and the data set.

2.1 The Algorithms

ID3 is a simple decision-tree learning algorithm developed by Ross Quinlan (1983; 1986b). It constructs a decision tree recursively, starting at the root. At each node, it selects, as

the feature to be tested at that node, the feature a_i whose mutual information with the output classification is greatest (this is sometimes called the information gain criterion). The training examples are then partitioned into those examples where $a_i = 0$ and those where $a_i = 1$. The algorithm is then invoked recursively on these two subsets of training examples. The algorithm halts when all examples at a node fall in the same class. At this point, a leaf node is created and labelled with the class in question. The basic operation of ID3 is quite similar to the CART algorithm developed by Breiman, Friedman, Olshen & Stone (1984) and to the tree-growing method developed by Lucassen and Mercer (1984).

In our implementation of ID3, we did not employ windowing (Quinlan, 1983), CHI-square forward pruning (Quinlan, 1986a), or any kind of reverse pruning (Quinlan, 1987). We did apply one simple kind of forward pruning to handle inconsistencies in the training data: If all training examples agreed on the value of the chosen feature, then growth of the tree was terminated in a leaf and the class having more training examples was chosen as the label for that leaf (in case of a tie, the leaf is assigned to class 0).

To apply ID3 to this task, the algorithm must be executed 26 times—once for each mapping h_i. Each of these executions produces a separate decision tree. The seven-letter window was represented as the concatenation of seven 29-bit strings. Each 29-bit string represents a letter (one bit for each letter, period, comma, and blank), and hence, only one bit is set to 1 in each 29-bit string. This produces a string of 203 bits for each window.

The error backpropagation algorithm (Rumelhart, Hinton & Williams, 1986) is widely applied to train artificial neural networks. We employed a fully-connected feed-forward network containing 203 input units, a single hidden layer of 160 units, and 26 output units (one for each mapping h_i). There were no shortcut connections from the input units to the output units. We employed the same input and output encodings described above.

Each artificial neuron is implemented by taking the dot product of a vector of weights $\mathbf{w}$ with a vector of incoming activations $\mathbf{x}$, adding a bias θ, and applying the logistic function

$$y = \frac{1}{1 + e^{-(\mathbf{w} \cdot \mathbf{x} + \theta)}},$$

which is a continuous, differentiable approximation to the linear threshold function used in perceptrons. The error backpropagation algorithm incrementally adjusts the weights and the bias of each unit in order to minimize the squared error between the computed and desired output values.

Unlike ID3, it is only necessary to apply BP once, because all 26 output bits can be learned simultaneously. Indeed, the 26 outputs all share the collection of 160 hidden units, which may allow them to be learned more accurately. However, while ID3 is a batch algorithm that processes the entire training set at once, BP is an incremental algorithm that makes repeated passes over the data. Each complete pass is called an "epoch." During an epoch, the training examples are inspected one-at-a-time, and the weights of the network are adjusted to reduce the squared error of the outputs. We used the implementation provided with (McClelland and Rumelhart, 1988).

As explained below, we concluded that the best performance was obtained with a learning rate of .25, a momentum coefficient of .9, random values in the range $[-.3, +.3]$, and a hidden layer containing 160 units. These values are identical to those employed by Sejnowski and Rosenberg, except that their network contained only 120 hidden units. As we will see below,

performance with 120 hidden units is not hugely different from performance with 160 hidden units.

Because the outputs from BP are floating point numbers between 0 and 1, we had to adapt the Hamming distance measure when mapping to the nearest legal phoneme/stress pair. We used the following distance measure: $d(\mathbf{x}, \mathbf{y}) = \sum_i |x_i - y_i|$. This reduces to the Hamming distance when $\mathbf{x}$ and $\mathbf{y}$ are Boolean vectors.

2.2 The Data Set

Sejnowski and Rosenberg provided us with a dictionary of 20,003 words and their corresponding phoneme and stress strings. From this dictionary we drew at random (and without replacement) a training set of 1000 words and a testing set of 1000 words.

2.3 Cross-validation Training

A serious impediment to the practical application of backpropagation is that it is controlled by many parameters including (a) the learning rate, (b) the momentum coefficient, (c) the random starting weights, (d) the range of values of the starting weights, (e) the target output activation level, (f) the number of units in the hidden layer, and (g) the criterion for halting training (the desired total sum squared error, TSSE). To determine good values for these parameters, we performed cross-validation training following the methodology described by Lang, Waibel, & Hinton (1990). To carry out this procedure, we split the training set further into an 800-word "subtraining" set and a 200-word "cross-validation" set. Then we executed the algorithm many times, varying these parameters while training on the subtraining set and testing on the cross-validation set. The goal of this search of parameter space is to find those parameters that give peak performance on the cross-validation set. These parameters can then be used to train on the entire training set and test on our 1000-word test set.

The advantage of cross-validation training is that no information from the test set is employed during training, and hence, the observed error rate on the test set is a better estimate of the true error rate of the learned network. This contrasts with the common, but unsound practice of adjusting parameters to optimize performance on the test set.

Preliminary cross-validation runs indicated that the following parameter values gave good performance: learning rate .25, momentum .9, range of initial random values $[-.3, +.3]$, target output activation levels 0.0 and 1.0. Other values that we tried were learning rate .1 or .2, initial weight range $[-.05, +.05]$, and target output activation levels 0.1 and 0.9. After these preliminary runs, we performed a series of runs that systematically varied the number of hidden units (40, 60, 80, 100, 120, 140, 160, and 180) and the random starting weights (four sets of random weights were generated for each network). Performance on the cross-validation set was evaluated after each complete pass through the training data (epoch). The networks were trained for 30 epochs (except for a few cases, where training was continued to 60 epochs to ensure that the peak performance had been found). Table 1 shows the peak performance (percent of letters correctly pronounced) for each network size and the total sum squared error (on the *subtraining set*) that gave the peak performance. These TSSE numbers (appropriately adjusted for the number of training examples) can then

Table 1: Optimal network size via cross-validation

Number of Hidden Units	Letters (% Correct)	TSSE	Number of Epochs
40	67.0	2289	28
60	67.7	939	46
80	68.3	1062	25
100	69.3	1041	19
120	68.7	1480	12
140	70.0	541	27
160	70.7	445	37
180	69.3	477	28

Table 2: Percent correct over 1000-word test set

	Level of Aggregation (% correct)				
Method	Word	Letter	Phoneme	Stress	Bit (mean)
ID3	9.6	65.6	78.7	77.2	96.1
BP	13.6**	70.6***	80.8***	81.3***	96.7*

Difference in the cell significant at $p < .05^{*}$, $.005^{**}$, $.001^{***}$

be used to decide when to terminate training on the entire training set. Based on these runs, the best network size is 160 hidden units.

Having completed cross-validation training, we then proceeded to train on the entire training set. During cross-validation training, we stored a snapshot of the weight values after the first complete epoch for each random network that was generated. Hence, to perform training on the entire training set, we used the best stored 160-hidden unit snapshot as a starting point. Our subtraining set contained 5,807 seven-letter windows, so to train our full training set (of 7,229 seven-letter windows), we placed the target TSSE at 554.

We were surprised by the figures shown in Table 1, since we expected that a reasonably small network (e.g., 80 hidden units) would give a good fit to the data. However, the table clearly shows that generalization steadily improves as the quality of the fit to the training data improves. Furthermore, Figure 1 shows that as training of a network continues past the point of peak performance, performance does not decline appreciably.

2.4 Results

Table 2 shows percent correct (over the 1000-word test set) for words, letters, phonemes, and stresses. Virtually every difference in the table at the word, letter, phoneme, and stress levels is statistically significant (using the test for the difference of two proportions). Hence, we conclude that there is a substantial difference in performance between ID3 and BP on this task.

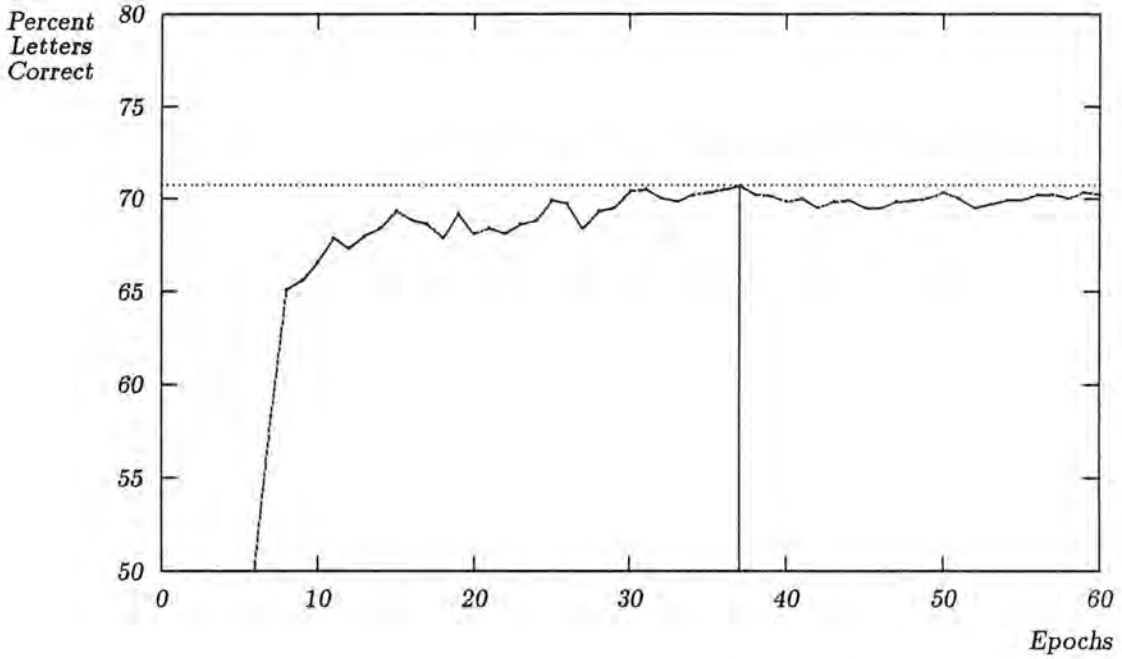

Figure 1: Training curve for the best 160-hidden unit network. Vertical bar indicates point of maximum performance.

Table 3: Classification of test set windows by ID3 and BP, decoding to nearest legal phoneme and stress.

		Back Propagation: Correct	Back Propagation: Incorrect	
ID3	Correct	4239 (58.5%)	512 (7.1%)	Disagree: 1385 (19.2%)
	Incorrect	873 (12.1%)	1618 (22.3%)	Agree: 5857 (80.9%)

It should be noted that although the testing set contains 1000 disjoint words, some of the 7-letter windows in the test set also appear in the training set. Specifically, 946 (13.1%) of the windows in the test set appear in the 1000-word training set. These represent 578 distinct windows. Hence, the performance at the letter, phoneme, and stress levels are all artificially high if one is concerned about the ability of the learning methods to correctly handle unseen cases. However, if one is interested in the probability that a letter (or phoneme, or stress) in an unseen word will be correctly classified, then these numbers provide the right measure.

To take a closer look at the performance difference, we can study exactly how each of the 7,242 seven-letter windows in the test set are handled by each of the algorithms. Table 3 categorizes each of these windows according to whether it was correctly classified by both algorithms, by only one of the algorithms, or by neither one.

The table shows that the windows correctly learned by BP do not form a superset of those learned by ID3. Instead, the two algorithms share 4,239 correct windows, and then each algorithm correctly classifies several windows that the other algorithm gets wrong. The net result is that BP classifies 361 more windows correctly than does ID3. This shows that the two algorithms, while they overlap substantially, have learned fairly different text-to-speech mappings.

The information in this table can be summarized as a correlation coefficient. Specifically, let X_{ID3} (X_{BP}) be a random variable that is 1 if and only if ID3 (BP, respectively) makes a correct prediction at the letter level. In this case, the correlation between X_{ID3} and X_{BP} is .5648. If all four cells of Table 3 were equal, the correlation coefficient would be zero.

A weakness of Table 2 is that it shows performance values for one particular choice of training and test sets. We have replicated this study four times (for a total of 5 independent trials). In each trial, we again randomly drew without replacement two sets of 1000 words from the dictionary of 20,003 words. Note that this means that there is some overlap among the five training sets (and among the five test sets). Table 4 shows the average performance of these 5 runs. All differences are significant below the .0001 level using a t-test for paired differences.

Throughout this paper, we will report results only for our first 1000-word training and test sets. Each of these results has been replicated on the four additional training and test sets, and the results of those replications are given in Appendix B.

In the remainder of this paper, we will attempt to understand the nature of the differences

Table 4: Average percent correct (1000-word test set) over five trials.

Method	Level of Aggregation (% correct)				
	Word	Letter	Phoneme	Stress	Bit (mean)
ID3	10.2	65.2	79.1	76.5	96.1
BP	15.1****	71.3****	81.3****	81.7****	96.7****

Difference in the cell significant at $p < .0001$****

between BP and ID3. Our main approach will be to experiment with modifications to the two algorithms that enhance or eliminate the differences between them (as measured by the correlation between their errors). All of these experiments are performed using only the training set and test set from Table 2.

3 Three Hypotheses

What causes the differences between ID3 and BP? We have three hypotheses:

Hypothesis 1: Overfitting. ID3 has overfit the training data, because it seeks complete consistency. This causes it to make more errors on the test set.

Hypothesis 2: Sharing. The ability of BP to share hidden units among all of the h_i allows it to reduce the aggregation problem at the bit level and hence perform better.

Hypothesis 3: Statistics. The numerical parameters in the BP network allow it to capture statistical information that is not captured by ID3.

These hypotheses are neither mutually exclusive nor exhaustive.

The following three subsections present the experiments that we performed to test these hypotheses.

3.1 Tests of Hypothesis 1 (Overfitting)

The tendency of ID3 to overfit the training data is well established in cases where the data contain noise. Three basic strategies have been developed for addressing this problem: (a) criteria for early termination of the tree-growing process, (b) techniques for pruning trees to remove overfitting branches, and (c) techniques for converting the decision tree to a collection of rules. We implemented and tested one method for each of these strategies. Table 5 summarizes the results.

The first row repeats the basic ID3 results given above, for comparison purposes. The second row shows the effect of applying a χ^2 test (at the .90 confidence level) to decide whether further growth of the decision tree is statistically justified (Quinlan, 1986a). As other authors have reported (Mooney et al., 1989), this hurts performance in the NETTALK domain. The third row shows the effect of applying Quinlan's technique of reduce-error pruning (Quinlan, 1987). Mingers (1989) provides evidence that this is one of the best pruning techniques. For this row, a decision tree was built using the 800-word subtraining

Table 5: Results of applying three overfitting-prevention techniques.

Method	Data set	Level of Aggregation (% correct)				
		Word	Letter	Phoneme	Stress	Bit (mean)
(a) ID3 (as above)	TEST:	9.6	65.6	78.7	77.2	96.1
(b) ID3 (χ^2 cutoff)	TEST:	9.1	64.8	78.4	77.1	96.1
(c) ID3 (pruning)	TEST:	9.3	62.4	76.9	75.1	95.8
(d) ID3 (rules)	TEST:	8.2	65.1	78.5	77.2	96.1

set and then pruned using the cross-validation set. Finally, the fourth row shows the effect of applying Quinlan's method for converting a decision tree to a collection of rules. Quinlan's method has three steps, of which we performed only the first two. First, each path from the root to a leaf is converted into a conjunctive rule. Second, each rule is evaluated to remove unnecessary conditions. Third, the rules are combined, and unnecessary rules are eliminated. The third step was too expensive to perform on this rule set, which contains 6,988 rules.

None of these techniques improved the performance of ID3 on this task. This suggests that Hypothesis 1 is incorrect: ID3 is not overfitting the data in this domain. This makes sense, since the only source of "noise" in this domain is the limited size of the 7-letter window and the existence of a small number of words like "read" that have more than one correct pronunciation. Seven-letter windows are sufficient to correctly classify 98.5% of the words in the 20,003-word dictionary. This may also explain why we did not observe overfitting during excessive training in our cross-validation runs with backpropagation either.

3.2 A Test of Hypothesis 2 (Sharing)

One obvious way to test the sharing hypothesis would be to develop a version of ID3 that permitted sharing among the 26 separate decision trees being learned. We could then see if this "shared-ID3" improved performance. An alternative is to *remove* sharing from back-propagation, by training 26 independent networks, each having only one output unit, to learn the 26 h_i mappings. If Hypothesis 2 is correct, then, because there is no sharing among these separate networks, we should see a drop in performance compared to the single network with shared hidden units. Furthermore, the decrease in performance should decrease the differences between BP and ID3.

For the single, shared-hidden-unit network, we employed a network with only 120 hidden units rather than the 160-hidden unit network described above. We employed 120 hidden units, because this experiment was conducted prior to the cross-validation study mentioned above, and because Sejnowski and Rosenberg (1987) employed a network with 120 hidden units, and we were replicating the parameters used by them. The performance of a 120-hidden unit network is comparable to the 160-hidden unit network (see Dietterich, Hild, and Bakiri, 1990).

There are two issues that arise in performing this experiment. The first issue concerns the number of hidden units to use in each of the 26 networks. If we try to learn each h_i mapping with a separate 120-hidden unit network, training does not converge. Hence, we

decided to try to find the smallest network, for each h_i, that would adequately fit the training examples.

The second issue concerns when to terminate training. Some of the 26 h_i are more difficult to learn than others. If we train each of the 26 separate networks to a single level of accuracy (e.g., 99% correct), the results cannot be directly compared to the performance of the single large network. To ensure the fairness of this comparison, we first measured the accuracy (over the training set) of each of the 26 bits learned by the single 120-hidden unit network. We expressed the accuracy as the sum of the squared error, since this is the criterion that is minimized by BP. We then trained each of the 26 separate networks to attain the squared error observed in the large network.

During training, the network (with a specified number H of hidden units) was trained for 240 epochs (or until the target error level was reached). If, after 240 epochs, the network was far away from the target error level and appeared to be stalled, we tried the next higher value for H. If the network was close to the target error level and still making progress, we continued training for an additional 240 epochs. If the network was close to the target error level but stalled, we restarted the training with a different, randomly chosen, setting of the weights.

Surprisingly, we were unable to successfully train the separate networks to the target error level on the 1000-word training set. We explored smaller subsets of the 1000-word training set (800, 400, 200, 100, and 50-words) and found that training did not succeed until the training set contained only 50 words! For the 100-word training set, for example, the individual networks often converged to local minima (even though the 120-hidden-unit network had avoided these minima). Specifically, bits 4, 6, 13, 15, 18, 21, and 25 could not be trained to criterion, even after 2000 epochs. This demonstrates that even if shared hidden units do not aid classification performance, they certainly aid the learning process!

As a consequence of this training problem, we are able to report results for only the 50-word training set. Table 3.2 summarizes the training process for each of the 26 output bits. Each row gives the number of hidden units, the squared error obtained from the 120-hidden-unit network, the squared error obtained from the individual network, and the number of epochs required for training. Notice that each individual bit was slightly over-trained as compared with the 120-hidden-unit network. This is because the program accumulates the squared errors during an epoch and stops when this falls below the target error level. Because performance improves during an epoch, the final squared error is somewhat lower.

Table 7 shows the performance of these 26 networks on the training and test sets. Performance on the training set is virtually identical to the 120-hidden-unit network, which shows that our training regime was successful. Performance on the test set, however, shows a loss of performance when there is no sharing of the hidden units among the output units. Hence, it suggests that Hypothesis 2 is at least partially correct.

However, examination of the correlation between ID3 and BP indicates that this is wrong. The correlation between X_{ID3} and X_{BP1} (i.e., BP on the single network) is .5167, whereas the correlation between X_{ID3} and X_{BP26} is .4942. Hence, the removal of shared hidden units has actually made ID3 and BP less similar, rather than more similar as Hypothesis 2 claims. This same result is obtained in the replications shown in Appendix B. The conclusion is that sharing in backpropagation is important to improving both its training and its performance, but it does not explain why ID3 and BP are performing differently.

Table 6: Training Statistics for 26 Independent Networks.

Bit	Number of hidden units	Squared error in 120-unit network	Squared error in separate network	Number of epochs
1	2	3.0735	2.995	41
2	2	1.1114	0.863	393
3	1	0.0378	0.037	48
4	2	0.1194	0.102	127
5	2	0.1183	0.107	311
6	1	0.0533	0.050	31
7	1	0.1076	0.094	33
8	1	0.0398	0.037	43
9	1	1.9828	1.978	9
10	1	0.0228	0.022	53
11	1	0.0309	0.030	40
12	1	0.0386	0.035	28
13	1	2.0893	2.043	57
14	1	0.0523	0.049	27
15	3	0.0724	0.066	313
16	2	0.4333	0.317	199
17	2	2.8737	2.479	123
18	3	0.0494	0.047	46
19	1	0.0010	0.002	60
20	1	0.0010	0.002	60
21	2	0.0520	0.050	167
22	2	0.0405	0.039	182
23	2	0.0528	0.044	253
24	1	0.9953	0.987	127
25	2	2.1431	2.080	36
26	1	0.0010	0.001	115

Table 7: Performance of 26 separate networks compared with a single network having 120 shared hidden units. Trained on 50-word training set, tested on 1000-word test set.

Method	Data set	Level of Aggregation (% correct)				
		Word	Letter	Phoneme	Stress	Bit (mean)
(a) ID3	TEST:	0.8	41.5	60.5	60.1	92.6
(b) BP 26 separate nets	TRAIN:	82.0	97.6	98.4	99.2	99.9
	TEST:	1.6	45.0	56.6	71.1	92.9
(c) BP 120 hidden units	TRAIN:	82.0	97.4	98.2	99.2	99.9
	TEST:	1.8	48.4	59.4	72.9	93.4
Difference (b)-(c)	TRAIN:	0.0	+0.2	+0.2	0.0	0.0
	TEST:	−0.2	−3.4***	−2.8**	−1.8*	−0.5
Difference (a)-(c)	TEST:	−1.0	−6.9	+1.1	−12.8	−0.9

Table 8: Performance of backpropagation with thresholded output values. Trained on 1000-word training set. Tested on 1000-word test set.

Method	Data set	Level of Aggregation (% correct)				
		Word	Letter	Phoneme	Stress	Bit (mean)
(a) ID3 (legal)	TEST:	9.6	65.6	78.7	77.2	96.1
(b) BP (legal)	TEST:	13.6**	70.6***	80.8***	81.3***	96.7*
(c) BP (thresholded)	TEST:	11.2	67.7	78.4	80.0	96.3
Difference (c)-(b)	TEST:	−2.4	−2.9***	−2.4***	−1.3*	−0.4

3.3 Tests of Hypothesis 3: Statistics

We performed three experiments to test the third hypothesis.

In the first experiment, we took the outputs of the back-propagation network and thresholded them (values $> .5$ were mapped to 1, values $\leq .5$ were mapped to 0) before mapping to the nearest legal phoneme/stress pair. Table 8 presents the results for the 1000-word training set.

The results show that thresholding significantly drops the performance of back-propagation. Indeed, at the phoneme level, the decrease is enough to push BP below ID3. At the other levels of aggregation, BP still out-performs ID3. These results support the hypothesis that the continuous outputs of the neural network aid the performance of BP.

However, thresholding the outputs of BP does not cause it to behave substantially more like ID3. The correlation between X_{ID3} and $X_{BPthresh}$ is .5685 (as compared with .5648 for X_{BP})—this is only a small increase. Close examination of the data shows that the 7-letter windows "lost" (i.e., incorrectly classified) when BP is thresholded include 120 windows correctly classified by ID3 and 112 windows incorrectly classified by ID3. Hence, the mistakes introduced by thresholding are nearly independent of the mistakes made by ID3.

While this experiment demonstrates the importance of continuous outputs, it does not tell us what kind of information is being captured by these continuous outputs nor does it reveal anything about the role of continuous weights inside the network. For this, we must

Table 9: Effect of "observed" decoding on learning performance.

Method	Data set	Level of Aggregation (% correct)				
		Word	Letter	Phoneme	Stress	Bit (mean)
(a) ID3 (legal)	TEST:	9.6	65.6	78.7	77.2	96.1
(b) BP (legal)	TEST:	13.6**	70.6***	80.8***	81.3***	96.7*
(c) ID3 (observed)	TEST:	13.0	70.1	81.5	79.2	96.4
(d) BP (observed)	TEST:	14.3	71.5*	82.0	81.4***	96.7
ID3 Improvement: (c)-(a)	TEST:	3.4***	4.5***	2.8***	2.0**	0.3
BP Improvement: (d)-(b)	TEST:	0.7	0.9	1.2*	0.1	0.0

turn to the other two experiments.

In the second experiment, we modified the method used to map a computed 26-bit string into one of the 324 strings representing legal phoneme/stress pairs. Instead of considering all possible legal phoneme/stress pairs, we restricted attention to those phoneme/stress pairs that had been observed in the training data. Specifically, we constructed a list of every phoneme/stress pair that appears in the training set (along with its frequency of occurrence). Appendix C shows this frequency information for the 1000-word training set. During testing, the 26-element vector produced either by ID3 or BP is mapped to the closest phoneme/stress pair appearing in this list. Ties are broken in favor of the most frequent phoneme/stress pair. We call this the "observed" decoding method, because it is sensitive to the phoneme/stress pairs (and frequencies) observed in the training set.

Table 9 presents the results for the 1000-word training set and compares them to the previous technique ("legal") that decoded to the nearest legal phoneme/stress pair. The key point to notice is that this decoding method leaves the performance of BP virtually unchanged while it substantially improves the performance of ID3. Indeed, it eliminates a substantial part of the difference between ID3 and BP—the two methods are now statistically indistinguishable at the word and phoneme levels. Mooney et al. (1989), in their comparative study of ID3 and BP on this same task, employed a version of this decoding technique (with random tie-breaking), and obtained very similar results when training on a set of the 808 words in the dictionary that occur most frequently in English text.

An examination of the correlation coefficients shows that "observed" decoding increases the similarity between ID3 and BP. The correlation between $X_{ID3observed}$ and $X_{BPobserved}$ is .5865 (as compared with .5648 for "legal" decoding). Furthermore, "observed" decoding is almost always monotonically better (i.e., windows incorrectly classified by "legal" decoding become correctly classified by "observed" decoding, but not vice versa).

From these results, we can conclude that BP was already capturing most of the information about the frequency of occurrence of phoneme/stress pairs, but that ID3 was not capturing nearly as much. Hence, this experiment strongly supports Hypothesis 3.

A drawback of the "observed" strategy is that it will never decode a window to a phoneme/stress pair that it has not seen before. Hence, it will certainly make some mistakes on the test set. However, phoneme/stress pairs that have not been observed in the training set make up a very small fraction of the windows in the test set. For example, only 7 of the phoneme/stress pairs that appear in our 1000-word test set do not appear in the 1000-word

Table 10: Effect of "block" decoding on learning performance.

Method	Data set	Level of Aggregation (% correct)				
		Word	Letter	Phoneme	Stress	Bit (mean)
(a) ID3 (legal)	TEST:	9.6	65.6	78.7	77.2	96.1
(b) BP (legal)	TEST:	13.6**	70.6***	80.8***	81.3***	96.7*
(c) ID3 (block)	TEST:	17.5	73.2	83.8	80.4	96.7
(d) BP (block)	TEST:	19.3	73.7	83.6	81.4	96.7
ID3 Improvement: (c)-(a)	TEST:	7.9***	7.6***	5.1***	3.2***	0.6*
BP Improvement: (d)-(b)	TEST:	5.7***	3.1***	2.8***	0.1	0.0

training set. In the test set, they only account for 11 of the 7,242 windows (0.15%). If we were to train on all 19,003 words from the dictionary that do not appear in our 1000-word test set, there would be only one phoneme/stress pair present in the test set that would not appear in the training set, and it would appear in only one window.

The final experiment concerning Hypothesis 3 focused on extracting additional statistical information from the training set. We were motivated by Klatt's (1987) view that ultimately letter-to-phoneme rules will need to identify and exploit morphemes (i.e., commonly-occurring letter sequences appearing within words). Therefore, we analyzed the training data to find all letter sequences of length 1, 2, 3, 4, and 5, and retained the 200 most-frequently-occurring sequences of each length. For each retained letter sequence, we formed a list of all phoneme/stress strings to which that sequence is mapped in the training set (and their frequencies). For example, here are the five pronunciations of the letter sequence "ATION" in the training set (Format is ($\langle$*phoneme string*$\rangle$ $\langle$*stress string*$\rangle$ $\langle$*frequency*$\rangle$)).

```
(("eS-xn" "1>0<<" 22)
 ("@S-xn" "1<0<<" 1)
 ("eS-xn" "2>0<<" 1)
 ("@S-xn" "2<0>>" 1)
 ("@S-xn" "1<0>>" 1))
```

During decoding, each word is scanned (from left to right) to see if it contains one of the "top 200" letter sequences of length k (varying k from 5 down to 1). If a word contains such a sequence, the letters corresponding to the sequence are processed as follows. First, each of the k windows centered on letters in the sequence is evaluated (i.e., by the 26 decision trees or by the feed-forward network) to obtain a 26-bit string, and these strings are concatenated to produce a bit string of length $k \cdot 26$. Then, each of the observed pronunciations for the sequence is converted into a $k \cdot 26$-bit string according to the code given in Appendix A. Finally, the "unknown" string is mapped to the nearest of these observed bit strings.

After decoding a block, control skips to the end of the matched k-letter sequence and resumes scanning for another "top 200" letter sequence of length k. After this scan is complete, the parts of the word that have not yet been matched are re-scanned to look for blocks of length $k-1$. Every letter in the word is eventually processed, because every individual letter is a block of length 1. We call this technique "block" decoding.

Table 11: Classification of test set windows by ID3 and BP with "block" decoding.

		Back Propagation: Correct	Back Propagation: Incorrect	
ID3	Correct	4892 (66.6%)	410 (5.7%)	Disagree: 855 (11.8%)
	Incorrect	445 (6.1%)	1495 (20.6%)	Agree: 6387 (88.2%)

Table 10 shows the performance results on the 1000-word test set. Block decoding significantly improves both ID3 and BP, but again, ID3 is improved much more (especially below the word level). Indeed, the two methods cannot be distinguished statistically at any level of aggregation. Furthermore, the correlation coefficient between $X_{ID3block}$ and $X_{BPblock}$ is .6973, which is a substantial increase compared to .5648 for legal decoding. Hence, block decoding also makes the performance of ID3 and BP much more similar. Table 11 shows how the 7,242 seven-letter windows of the test set are handled by ID3 and BP.

Curiously, these summary numbers hide substantial shifts in performance caused by block decoding. To demonstrate this, consider that there is only a .7153 correlation between $X_{ID3legal}$ and $X_{ID3block}$. This reflects the fact that while "block" decoding gains 734 windows previously misclassified by "legal" decoding, it also loses 183 windows that were previously correctly classified by "legal" decoding. Similarly, there is only a .7693 correlation between $X_{BPlegal}$ and $X_{BPblock}$ (reflecting a gain of 452 and a loss of 227 windows).

The conclusion we draw is that block decoding further reduces the differences between ID3 and BP, and hence that this experiment also supports Hypothesis 3. The experiment suggests that the block decoding technique is a useful adjunct to any learning algorithm applied in this domain. It also suggests that the performance of block decoding could be improved if some way could be found to avoid losing windows that were correctly classified without block decoding. One technique we are exploring is to combine the constraints of blocks that overlap.

4 Discussion

4.1 Improving These Algorithms

The results shown in previous sections demonstrate that ID3 and BP, while they attain similar levels of performance, still do not cover the *same* set of testing examples. In particular, an analysis of the 7,242 7-letter windows in the test set reveals that there are 855 windows that are incorrectly classified by one of the algorithms and correctly classified by the other. This suggests that an inductive learning algorithm should be able to label correctly all of these 855 windows. This would yield a performance of 79.4% at the letter level, which would be quite good.

Table 12: Best configuration: ID3, 15-letter window, 127-bit error correcting code, 7-letter phoneme and stress context, domain-specific input features, observed decoding, simplified stresses.

	Level of Aggregation (% correct)				
Training set	Word	Letter	Phoneme	Stress	Bit (mean)
1,000 words	40.6	84.1	87.0	91.4	92.2
19,003 words	64.8	91.4	93.7	95.1	95.8

There are several directions that can be explored for improving these algorithms. We have pursued several of these directions in order to develop a high-performance text-to-speech system. Our efforts are reported elsewhere (Bakiri & Dietterich, 1990b).

One approach is to design better output codes for phoneme/stress pairs. Our experiments have shown that BCH error correcting codes provide better output codes than the output code used in this paper. Randomly-generated bit-strings produce similar performance improvements (see Bakiri & Dietterich, 1990a).

Another approach is to widen the 7-letter window and introduce context. Lucassen and Mercer (1984) employ a 9-letter window and scan the word from back-to-front. They also include the phonemes and stresses of the four letters to the right of the letter at the center of the window. These phonemes and stresses can be obtained, during execution, from the letters that have already been pronounced during the scan. Our experiments (with a 15-letter window) indicate that this produces substantial performance gains as well.

A third technique for improving performance is to supply additional input features to the program. One feature of letters that helps is a bit indicating whether the letter is a vowel or a consonant. A feature of phonemes that helps is whether the phoneme is tense or lax.

A fourth technique to be pursued is to refine the block decoding method. Blocks should be chosen more carefully and checked by cross validation. Decoding should consider overlapping blocks.

A fifth direction that we have not yet pursued is to implement one of the published methods for obtaining class probability estimates from decision trees. Buntine (1990), for example, presents an algorithm that provides fairly accurate probability estimates at the leaves of a decision tree, rather than the simple binary outputs that we employed. This could eliminate the need for "observed" decoding.

By combining the error-correcting output codes with a wider window, a back-to-front scan to include phoneme and stress context, and domain specific features, we have obtained excellent performance with our 1000-word training and test sets. Table 12 shows our best-performing configuration when trained on 1000 words and when trained on 19,003 words. Details of this configuration are described in Bakiri & Dietterich (1990b).

It is difficult to compare these performance levels to the human-developed letter-to-sound rules incorporated in systems such as DECtalk (Klatt, 1987). The criteria for correctness employed in this study are very strict: The phoneme and stress classes must match exactly. A little thought shows that this is overly strict. Some of the stress symbols (i.e., < and >) do not affect pronunciation at all. Other stress symbols (i.e., secondary, 2, and tertiary, 0,

stresses) are indistinguishable when played through the DECtalk synthesizer. Some phoneme errors (e.g., substituting /k/ for /e/) are very serious, while others (e.g., substituting /x/ for /@/) are virtually indistinguishable. Hence, additional research is required to develop an error cost function that would permit better comparisons. Without such a cost function, the comparisons in this paper among learning algorithms are still valuable, because they provide a difficult, large scale classification problem. But the performance figures shown here do not give a good indication of the quality of the speech produced by applying the learned classification trees or networks.

Klatt (1987) points out three properties of the domain that present special challenges to inductive learning methods:

> (1) the considerable extent of letter context that can influence stress patterns in a long word (and hence affect vowel quality in words like "photograph/photography"), (2) the confusion caused by some letter pairs, like CH, which function as a single letter in a deep sense, and thus misalign any relevant letters occurring further from the vowel, and (3) the difficulty of dealing with compound words (such as "houseboat" with its silent "e"), i.e., compounds act as if a space were hidden between two of the letters inside the word.

Long-distance interactions pose a difficult problem for BP, since capturing them presumably requires a very wide window. This in turn requires a very large network with many weights, and these will be much more difficult and time-consuming to train. ID3 scales very well as the number of irrelevant features grows, so we have been able to apply it to much wider windows without problems. General solutions to the other two problems mentioned by Klatt appear to be quite challenging.

4.2 Applying ID3 to Aid BP

An interesting observation from this and other studies is that the performance of ID3 and BP is highly correlated. This suggests a methodology for using ID3 to aid BP even in domains where BP out-performs ID3. In many real-world applications of inductive learning, substantial "vocabulary engineering" is required in order to attain high performance. This vocabulary engineering process typically involves the iterative selection and testing of promising features. To test the features, it is necessary to train a BP network using them—which is very time-consuming. Because ID3 is so highly correlated with BP, it could be used instead to test feature sets. Once a good set of features is identified, a BP network could then be trained.

To examine this idea in more detail, consider Table 13. This shows the performance of ID3 and BP on each of the 26 individual bits (i.e., without decoding them at all). (Each algorithm was trained on the 1000-word training set and tested on the 1000-word test set. A 120-hidden unit network was employed with BP). The correlation coefficient is .9812, which is significant well below the .001 level.

In addition to considering the similarity in the predictive performance of ID3 and BP, we can also consider the difficulty of training the two algorithms. Table 14 shows the performance of ID3 and BP on the individual output bits when trained on the 50-word

Table 13: Performance, complexity, and difficulty of learning. 50-word training set, 1000-word test set.

	ID3		BP	
bit	windows	(%)	windows	(%)
1	6984	96.4	6963	96.1
2	6779	93.6	6781	93.6
3	7104	98.1	7110	98.2
4	6936	95.8	6925	95.6
5	6584	90.9	6591	91.0
6	7065	97.6	7064	97.5
7	7207	99.5	7181	99.2
8	7213	99.6	7198	99.4
9	7206	99.5	7214	99.6
10	7237	99.9	7232	99.9
11	7240	100.0	7239	100.0
12	7202	99.4	7158	98.8
13	6810	94.0	6801	93.9
14	7148	98.7	7138	98.6
15	6944	95.9	6914	95.5
16	6903	95.3	6966	96.2
17	6629	91.5	6640	91.7
18	6863	94.8	6997	96.6
19	7242	100.0	7242	100.0
20	7242	100.0	7242	100.0
21	6863	94.8	6997	96.6
22	6658	91.9	6752	93.2
23	6682	92.3	6779	93.6
24	6542	90.3	6603	91.2
25	6729	92.9	6798	93.9
26	7242	100.0	7242	100.0

training set (and tested on the 1000-word test set). For BP, these are the 26 separate, individually-sized networks that were trained to test Hypothesis 2. The table also shows some measures of the difficulty of learning. For ID3, the number of nodes in the learned decision tree and the depth of the tree are given. For BP, the minimum number of hidden units and the number of epochs are shown. As with Table 13, the performance is highly correlated ($r = .9550$). More importantly, the number of nodes and the number of hidden units have a correlation coefficient of .6602. There is a reasonably strong correlation between the number of nodes and the number of epochs as well (0.5913).

This shows that ID3 and BP tend to agree on the difficulty of these learning problems. This agreement suggests that by running ID3 on a new problem, we could obtain an estimate of the number of hidden units needed by BP and the number of epochs required to train them. This would be very useful, because it could reduce the number of cross-validation runs required to determine good parameters for BP.

5 Conclusions

The relative performance of ID3 and Backpropagation on the text-to-speech task depends on the decoding technique employed to convert the 26 bits of the Sejnowski/Rosenberg code into phoneme/stress pairs. Decoding to the nearest legal phoneme/stress pair (the most obvious approach) reveals a substantial difference in the performance of the two algorithms.

Experiments investigated three hypotheses concerning the cause of this performance difference.

The first hypothesis—that ID3 was overfitting the training data—was shown to be incorrect. Three techniques that avoid overfitting were applied, and none of them improved ID3's performance.

The second hypothesis—that the ability of back propagation to share hidden units was a factor—was shown to be only partially correct. Sharing of hidden units does improve the classification performance of backpropagation and—perhaps more importantly—the convergence rate of the gradient descent search. However, an analysis of the kinds of errors made by ID3 and backpropagation (with or without shared hidden units) demonstrated that these were different kinds of errors. Hence, eliminating shared hidden units does not produce an algorithm that behaves like ID3. This suggests that the development of a "shared ID3" algorithm that could learn multiple concepts simultaneously is unlikely to produce performance similar to BP.

The third hypothesis—that backpropagation was capturing statistical information by some mechanism (perhaps the continuous output activations)—was demonstrated to be the primary difference between ID3 and BP. By adding the "observed" decoding technique to both algorithms, the level of performance of the two algorithms in classifying test cases becomes statistically indistinguishable (at the word and phoneme levels). By adding the "block" decoding technique, all differences between the algorithms are statistically insignificant.

Given that with block decoding the two algorithms perform equivalently, and given that BP is much more awkward to apply and time-consuming to train, these results suggest that in tasks similar to the text-to-speech task, ID3 with block decoding is clearly the algorithm of

Table 14: Performance, complexity, and difficulty of learning. 50-word training set, 1000-word test set.

	ID3			BP		
bit	windows (%)	nodes	depth	windows (%)	# hidden units	epochs
1	6743 (93.1)	33	14	6465 (89.3)	2	41
2	6388 (88.2)	44	18	6372 (88.0)	2	393
3	7008 (96.8)	9	5	6973 (96.3)	1	48
4	6577 (90.8)	38	14	6448 (89.0)	2	127
5	5977 (82.5)	57	16	5833 (80.5)	2	311
6	6734 (93.0)	18	8	6729 (92.9)	1	31
7	7154 (98.8)	7	4	7116 (98.3)	1	33
8	7029 (97.1)	14	9	6963 (96.1)	1	43
9	7163 (98.9)	3	2	7143 (98.6)	1	9
10	7224 (99.8)	4	3	7209 (99.5)	1	53
11	7241 (100.0)	5	3	7221 (99.7)	1	40
12	6878 (95.0)	16	8	6798 (93.9)	1	28
13	6500 (89.8)	30	9	6408 (88.5)	1	57
14	7057 (97.4)	18	11	6919 (95.5)	1	27
15	6624 (91.5)	25	9	6518 (90.0)	3	313
16	6781 (93.6)	26	10	6829 (94.3)	2	199
17	6247 (86.3)	45	12	6133 (84.7)	2	123
18	6438 (88.9)	35	14	6363 (87.9)	3	46
19	7242 (100.0)	1	0	7242 (100.0)	1	60
20	7242 (100.0)	1	0	7242 (100.0)	1	60
21	6438 (88.9)	35	14	6199 (85.6)	2	167
22	5904 (81.5)	48	12	6308 (87.1)	2	182
23	6302 (87.0)	41	27	6390 (88.2)	2	253
24	6208 (85.7)	42	11	6281 (86.7)	1	127
25	6517 (90.0)	40	12	6413 (88.6)	2	36
26	7242 (100.0)	1	0	7242 (100.0)	1	115

choice. For other applications of BP, ID3 can play an extremely valuable role in exploratory studies to determine good sets of features and predict the difficulty of learning tasks.

6 Acknowledgements

The authors thank Terry Sejnowski for providing the Nettalk phonemic dictionary, without which this work would have been impossible. Correspondence with Jude Shavlik, Ray Mooney, and Geoffrey Towell helped clarify the possible kinds of decoding strategies. Discussions with Lorien Pratt aided in the design of the cross-validation studies. This research was supported by NSF grant numbers CCR-87-16748 and IRI-86-57316 (Presidential Young Investigator Award) with matching support from SUN Microsystems.

7 References

Bakiri, G., & Dietterich, T. G. (1990a). Applying machine learning techniques to construct high-performance letter-to-sound rules for English. Forthcoming.

Bakiri, G., & Dietterich, T. G. (1990b). Boosting the performance of inductive learning programs via error-correcting output codes. Forthcoming.

Breiman, L, Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984). *Classification and Regression Trees.* Monterey, CA: Wadsworth and Brooks.

Buntine, W. (1990). A theory of learning classification rules. Doctoral dissertation. University of Technology, School of Computing Science, Sydney.

Dietterich, T. G., Hild, H., & Bakiri, G. (1990). A comparative study of ID3 and Backpropagation for English Text-to-Speech Mapping. *Proceedings of the Seventh International Conference on Machine Learning* (pp. 24–31). Austin, TX: Morgan Kaufmann.

Klatt, D. (1987). Review of text-to-speech conversion for English. *J. Acoust. Soc. Am., 82,* (3), 737–793.

Lucassen, J. M., and Mercer, R. L. (1984). An information theoretic approach to the automatic determination of phonemic base forms. Proc. Int. Conf. Acoust. Speech Signal Process. ICASSP-84, 42.5.1–42.5.4.

Lang, K. J., Waibel, A. H., and Hinton, G. E. (1990). A time-delay neural network architecture for isolated word recognition. *Neural Networks, 3,* 33-43.

McClelland, J. L., and Rumelhart, D. E. (1988). *Explorations in Parallel Distributed Processing,* Cambridge, MA: MIT Press.

Mingers, J. (1989). An empirical comparison of pruning methods for decision tree induction. *Machine Learning, 4* (2), 227–243.

Mooney, R., Shavlik, J., Towell, G., and Gove, A. (1989). An experimental comparison of symbolic and connectionist learning algorithms. *IJCAI-89: Eleventh International Joint Conference on Artificial Intelligence.* 775–80.

Quinlan, J. R. (1983). Learning efficient classification procedures and their application to chess endgames, in Michalski, R. S., Carbonell, J., and Mitchell, T. M., (eds.), *Machine learning: An artificial intelligence approach, Vol. I*, Palo Alto: Tioga Press. 463–482.

Quinlan, J. R. (1986a). The effect of noise on concept learning. In Michalski, R. S., Carbonell, J., and Mitchell, T. M., (eds.), *Machine learning, Vol. II*, Palo Alto: Tioga Press. 149–166.

Quinlan, J. R. (1986b). Induction of Decision Trees, *Machine Learning,1* (1), 81–106.

Quinlan, J. R., (1987). Simplifying decision trees. *International Journal of Man-Machine Studies, 27*, 221–234.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning internal representations by error propagation. In Rumelhart, D. E., and McClelland, J. L., (eds.) *Parallel Distributed Processing*, Vol 1. 318–362.

Sejnowski, T. J., and Rosenberg, C. R. (1987). Parallel networks that learn to pronouce English text. *Complex Systems, 1*, 145–168.

A The 26-bit Code for Phoneme/Stress Pairs

Sejnowski and Rosenberg developed the following distributed code for representing the phonemes and stresses. The examples were supplied with their database.

Phoneme Code		
Phoneme	Codeword	Examples
/a/	00001000000100100000	wAd, dOt, Odd
/b/	00010000001010000000	Bad
/c/	00001000000000010000	Or, cAUght
/d/	10000000001010000000	aDd
/e/	01000000000100010000	Angel, blAde, wAy
/f/	00010001000000000000	Farm
/g/	00000100001010000000	Gap
/h/	00100000100000000000	Hot, WHo
/i/	00010000000101000000	Eve, bEe
/k/	00000100001000000000	Cab, Keep
/l/	01000000100010000000	Lad
/m/	00010000010010000000	Man, iMp
/n/	10000000010010000000	GNat, aNd
/o/	00100000000100010000	Only, Own
/p/	00010000001000000000	Pad, aPt
/r/	00001000100010000000	Rap
/s/	10000010000000000000	Cent, aSk
/t/	10000000001000000000	Tab
/u/	00100000000101000000	bOOt, OOze, yOU
/v/	00010001000010000000	Vat
/w/	00010000100010000000	We, liqUid
/x/	00001000000000010000	pirAte, welcOme
/y/	00001000100010000000	Yes, senIor
/z/	10000010000010000000	Zoo, goeS
/A/	11000000000100010000	Ice, hEIght, EYe
/C/	00001010000000000000	CHart, Cello
/D/	01000010000010000000	THe, moTHer
/E/	01010000000000010000	mAny, End, hEAd
/G/	00000100001001000000	leNGth, loNG, baNk
/I/	00010000000001000000	gIve, bUsy, captAIn
/J/	00001010000010000000	Jam, Gem
/K/	00001111000000000000	aNXious, seXual
/L/	10000000100010000000	eviL, abLe
/M/	01000000010010000000	chasM
/N/	00001000010010000000	shorteN, basiN
/O/	10001000000100010000	OIl, bOY

Phoneme Code		
Phoneme	Codeword	Examples
/Q/	000101100001010000000	Quilt
/R/	000001000100010000000	honeR, afteR, satyR
/S/	000010010000000000000	oCean, wiSH
/T/	010000010000000000000	THaw, baTH
/U/	000001000000001000000	wOOd, cOUld, pUt
/W/	000011000000101010000	oUT, toWel, hoUse
/X/	110000100000000000000	miXture, anneX
/Y/	110100000000101000000	Use, fEUd, nEw
/Z/	000010010000010000000	uSual, viSion
/@/	010000000000000100000	cAb, plAId
/!/	010100100000000000000	naZi, piZZa
/#/	000011100000010000000	auXiliary, eXist
/*/	100100001000010100000	WHat
/^/	100000000000000100000	Up, sOn, blOOd
/+/	000000000000000000000	abattOIr, mademOIselle
/-/	000000000000000001001	silence
/_/	000000000000000001010	word-boundary
/./	00000000000000000110	period

Here are the meanings of the individual bit positions:

Bit Position	Meaning
1	Alveolar = Central1
2	Dental = Front2
3	Glottal = Back2
4	Labial = Front1
5	Palatal = Central2
6	Velar = Back1
7	Affricative
8	Fricative
9	Glide
10	Liquid
11	Nasal
12	Stop
13	Tensed
14	Voiced
15	High
16	Low
17	Medium
18	Elide
19	FullStop
20	Pause
21	Silent

The stress code actually encodes syllable boundary information as well as stresses.

Stress Code		
Stress	Codeword	Meaning
<	10000	a consonant or vowel following the first vowel of the syllable nucleus.
>	01000	a consonant prior to a syllable nucleus.
0	00010	the first vowel in the nucleus of an unstressed syllable.
2	00100	the first vowel in the nucleus of a syllable receiving secondary stress.
1	00110	the first vowel in the nucleus of a syllable receiving primary stress.
-	11001	silence

Table 15: Percent correct over 1000-word test set

Dataset	Method	Level of Aggregation (% correct)				
		Word	Letter	Phoneme	Stress	Bit (mean)
a	ID3	9.6	65.6	78.7	77.2	96.1
	BP	13.6**	70.6***	80.8***	81.3***	96.7*
b	ID3	10.4	65.6	79.6	76.4	96.1
	BP	15.7***	71.5***	81.7***	81.4***	96.7*
c	ID3	10.5	64.4	78.9	75.7	96.0
	BP	15.2***	71.4***	81.4***	81.7***	96.7*
d	ID3	10.9	65.8	80.0	76.2	96.2
	BP	16.3***	71.3***	81.4*	81.6***	96.7*
e	ID3	9.5	64.7	78.2	77.1	96.0
	BP	14.5**	71.6***	81.3***	82.3***	96.7*

Difference in the cell significant at $p < .05^*$, $.005^{**}$, $.001^{***}$

Table 16: Correlation Between ID3 and BP

Dataset	Legal	Observed	Block
a	.5648	.5865	.6973
b	.5844	.5945	.7110
c	.5593	.5796	.7077
d	.5722	.5706	.6723
e	.5563	.5738	.6879
Average Increase		.0136	.1278

B Replication of Results on Four Additional Data Sets

To simplify the presentation in the body of the paper, we presented data only for one choice of training and test sets. This appendix provides that same data on all five training and testing sets to demonstrate that the results hold in general. Table 15 shows the performance, under legal decoding, of ID3 and BP when trained on each of the 5 training sets and tested on the corresponding test sets.

B.1 Performance of ID3 and BP under legal decoding

Table 16 shows the correlation coefficients between the errors made by ID3 and by BP under the three different decoding methods. Notice that, with one exception, observed decoding always enhances the similarity between ID3 and BP, and that block decoding substantially increases this similarity. The "average increase" row shows the average increase in the correlation coefficient over legal decoding.

Table 17: Performance difference (in percentage points) between a single 120-hidden unit network and 26 seperate networks. Trained on 50 words and tested on 1000 words.

Replication	Data set	Level of Aggregation (% point differences)				
		Word	Letter	Phoneme	Stress	Bit (mean)
a	TRAIN:	0.0	−0.2	−0.2	0.0	0.0
	TEST:	0.2	3.4	2.8	1.8	0.6
b	TRAIN:	4.0	0.1	−0.2	−0.5	−0.1
	TEST:	0.6	3.4	2.9	0.8	−0.2
c	TRAIN:	4.0	0.0	0.0	0.0	−0.1
	TEST:	1.1	3.0	2.2	2.9	0.0
d	TRAIN:	0.0	−0.6	−0.5	0.0	−0.1
	TEST:	0.3	3.7	2.8	2.1	0.0
e	TRAIN:	4.0	−0.5	−0.6	0.0	−0.1
	TEST:	0.7	2.7	2.1	1.6	−0.1
Averages	TRAIN:	2.4	−0.2	−0.2	0.1	−0.1
	TEST:	0.6	3.2	2.6	1.8	0.1

Table 18: Error Correlations

Replication	Correlation Coefficients	
	X_{ID3} and X_{BP1}	X_{ID3} and X_{BP26}
a	.5167	.4942
b	.5005	.4899
c	.5347	.5062
d	.4934	.4653
e	.4934	.4790
Average Decrease		0.0208

B.2 Tests of the Sharing Hypothesis

For replications b, c, d, and e, the training procedure for each of the 26 separate networks was slightly different from the procedure described for replication a. Starting with $H = 1$, a network with H hidden units was trained for 1000 epochs. If this did not attain the desired fit with the data, the next larger value for H was tried. If a network with 5 hidden units failed to fit the data, the process was repeated, starting again with $H = 1$ and a new randomly-initialized network. No network required more than 4 hidden units. Table 17 shows the observed performance differences. The training figures show that, with the exception of word-level and stress-level performance, the 26 separate nets fit the training data slightly better than the single 120-hidden-unit network.

C Frequency Information for Phoneme/Stress Pairs Observed in the 1000-Word Training Set

Phoneme/stress pairs observed in the 1000-word training set.

Phoneme	Stress	Count		Phoneme	Stress	Count
a	<	3		p	<	58
a	0	15		p	>	165
a	1	97		r	<	139
a	2	21		r	>	230
b	<	32		s	<	158
b	>	121		s	>	188
c	<	2		t	<	249
c	0	2		t	>	173
c	1	39		t	0	1
c	2	15		u	<	2
d	<	99		u	>	1
d	>	97		u	0	8
e	<	3		u	1	30
e	0	4		u	2	5
e	1	86		v	<	39
e	2	38		v	>	53
f	<	17		w	>	37
f	>	81		w	0	4
g	<	26		w	1	10
g	>	57		w	2	3
h	>	31		x	<	57
i	<	2		x	>	4
i	0	144		x	0	515
i	1	51		x	1	3
i	2	16		x	2	17
k	<	114		y	>	5
k	>	201		y	0	6
l	<	108		y	1	2
l	>	163		z	<	48
m	<	95		z	>	9
m	>	107		A	<	2
n	<	382		A	0	12
n	>	81		A	1	45
o	0	17		A	2	31
o	1	67		C	<	13
o	2	26		C	>	22

Phoneme	Stress	Count		Phoneme	Stress	Count
D	<	7		Z	<	6
D	>	2		Z	>	5
E	<	2		@	<	1
E	>	1		@	0	15
E	0	3		@	1	164
E	1	138		@	2	35
E	2	35		!	>	1
G	<	37		#	<	1
I	<	5		*	>	4
I	0	151		^	0	1
I	1	133		^	1	54
I	2	43		^	2	4
J	<	21		+	1	1
J	>	40		-	<	542
K	<	1		-	>	233
L	<	54		-	0	241
L	>	38		-	1	34
M	0	11		-	2	8
N	<	14				
O	1	6				
O	2	2				
R	<	119				
S	<	27				
S	>	49				
T	<	5				
T	>	18				
U	0	5				
U	1	12				
U	2	7				
W	0	1				
W	1	17				
W	2	4				
X	<	14				
Y	0	20				
Y	1	25				
Y	2	8				