A Comparative Study 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 97331-3902

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 out-performs 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 (see Klatt, 1987). One particularly difficult step involves mapping words (i.e., strings of letters) into strings of phonemes and stresses. In this paper, we compare two machine learning algorithms applied to the task of learning this text-to-speech mapping. We employ the formulation developed by Sejnowski and Rosenberg (1987) in their widely known work on NETTALK.

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 $f : L^* \longrightarrow P^* \times S^*$. 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, 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). The version we employed used the information gain criterion to choose which feature to place at the root of each decision tree (and subtree). 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 replicated the network architecture and training procedure employed by Sejnowski and Rosenberg (1987). This network is a fully-connected feed-forward network containing 203 input units, 120 hidden units, and 26 output units (one for each mapping h_i). We employed the same input and output encodings described above.

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 120 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 a learning rate of .25 and a momentum term of .9. The weights of the network were initialized to random values between $-.3$ and $+.3$. In all cases, we trained for 30 epochs, since this was the training regime followed by Sejnowski and Rosenberg. We used the implementation provided with (McClelland and Rumelhart, 1988).

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. This dictionary was randomly partitioned into a testing set of 1000 words, and a training set of 19,003 words. This training set was further subdivided to extract smaller training sets of 1000, 800, 400, 200, 100, and 50 words. Each smaller training set was extracted by randomly sampling from the next larger set.

2.3 Results

Table 1 shows percent correct (over the 1000-word test set) as a function of the size of the training 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.

To take a closer look at the performance difference, we can study exactly how each of the 7,242 windows in the test set are handled by each of the algorithms. Table 2 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.

Table 1: Percent correct over 1000-word test set

Sample Size	Method	Level of Aggregation (% correct)				
		Word	Letter	Phoneme	Stress	Bit (mean)
50	ID3	0.8	41.5	60.5	60.1	93.1
	BP	1.8*	48.4***	59.4	72.9***	93.5
100	ID3	2.0	47.3	64.1	65.8	94.0
	BP	3.7*	55.2**	66.1**	75.5**	94.4
200	ID3	4.4	56.6	70.5	72.2	95.1
	BP	6.0	61.4***	71.9*	78.6***	95.3
400	ID3	6.2	58.7	73.7	72.1	95.5
	BP	10.5***	65.7***	76.0***	79.9***	95.9
800	ID3	9.6	63.8	77.8	75.6	96.2
	BP	12.2*	68.7***	78.9	80.7***	96.3
1000	ID3	9.6	65.6	78.7	77.2	96.4
	BP	14.7***	70.9***	81.1***	81.4***	96.6

Difference in the cell significant at $p < .05^{*}$, $.01^{**}$, $.001^{***}$

Table 2: Classification of test set windows by ID3 and Backpropagation, decoding to nearest legal phoneme and stress.

		Back Propagation		
		Correct	Incorrect	
ID3	Correct	4231	520	4751
	Incorrect	907	1583	2490
		5138	2103	

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 4,231 correct windows, and then each algorithm correctly classifies several windows that the other algorithm gets wrong. The net result is that BP classifies 387 more windows correctly than does ID3. This shows that the two algorithms, while they share substantial overlap, have learned substantially 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 iff ID3 (BP, respectively) makes a correct prediction at the letter level. In this case, the correlation between X_{ID3} and X_{BP} is .5508. If all four cells of Table 2 were equal, the correlation coefficient would be zero.

A weakness of Table 1 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). Table 3 shows the average performance of these 5 runs (each, of course, on a different randomly-drawn 1000-word test set). All differences are significant below the .0001 level using a t-test for paired differences.

In the remainder of this paper, we will attempt to understand the nature of the differences 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. All of these experiments are performed using only the training set and test set from Table 1.

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 may allow it to reduce the aggregation problem at the bit level.

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 exclusive nor exhaustive.

The following two 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 4 summarizes the results.

Table 3: Average percent correct (1000-word test set) over five trials.

Sample Size	Method	Level of Aggregation (% correct)				
		Word	Letter	Phoneme	Stress	Bit (mean)
1000	ID3	10.2	65.2	79.1	76.5	96.4
	BP	14.3*	70.7*	81.2*	81.0*	96.6*

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

Table 4: 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	96.4
(b) ID3 (χ^2 cutoff)	TEST:	9.1	64.8	78.4	77.1	96.4
(c) ID3 (pruning)	TEST:	9.3	62.4	76.9	75.1	96.1
(d) ID3 (rules)	TEST:	8.2	65.1	78.5	77.2	96.4

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 training set, and then pruned using the additional words from the 1000-word training set that do not appear in the 800-word training set. Other trials using words from a 1600-word training set produced similar results. 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.

3.2 A Test of Hypothesis 2 (Sharing)

To test the sharing hypothesis, we attempted to train 26 independent networks, each having only one output unit, to learn the 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.

Surprisingly, we were unable to train successfully the separate networks to the target error level on any training set other than the 50-word set. 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). This shows 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 5 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 suggests. The conclusion is that sharing in backpropagation is important to improving its performance, but it does not explain why ID3 and BP are performing differently.

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

Table 5: 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.

		Level of Aggregation (% correct)				
Method	Data set	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	93.5
Difference (b)-(c)	TRAIN:	0.0	+0.2	+0.2	0.0	0.0
	TEST:	-0.2	-3.4***	-2.8**	-1.8*	-0.6
Difference (a)-(c)	TEST:	-1.0	-6.9	+1.1	-12.8	-0.9

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 6 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. However, at the other levels of aggregation, BP still out-performs ID3. Nevertheless, the results support the hypothesis that the continuous outputs of the neural network aid the performance of BP. A comparison of correlation coefficients confirms this. The correlation between X_{ID3} and $X_{BPthresh}$ is .5598 (as compared with .5508 for X_{BP}).

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 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). During testing, the 26-bit 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 7 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. Mooney et al. (1989), in their comparative study of ID3 and BP on this same task, employed a version of this decoding technique (without the tie-breaking by frequency), 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 .5705 (as compared with .5508 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.

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 (($\langle phonemestring \rangle$ $\langle stressstring \rangle$ $\langle frequency \rangle$))).

```
(("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, it is mapped and decoded as follows. First, each of the k windows in the

Table 6: Performance of backpropagation with thresholded output values. Trained on 1000-word training set. Tested on 1000-word test set.

		Level of Aggregation (% correct)				
Method	Data set	Word	Letter	Phoneme	Stress	Bit (mean)
(a) ID3 (legal)	TEST:	9.6	65.6	78.7	77.2	96.1
(b) BP (legal)	TEST:	14.7	70.9	81.1	81.4	96.6
(c) BP (thresholded)	TEST:	12.1	67.9	78.6	80.3	96.6
Difference (c)-(b)	TEST:	-2.6***	-3.0***	-2.5***	-1.1*	0.0

Table 7: Effect of "observed" decoding on learning performance.

		Level of Aggregation (% correct)				
Method	Data set	Word	Letter	Phoneme	Stress	Bit (mean)
(a) ID3 (legal)	TEST:	9.6	65.6	78.7	77.2	96.1
(b) BP (legal)	TEST:	14.7***	70.9***	81.1***	81.4***	96.6
(c) ID3 (observed)	TEST:	13.0	70.1	81.5	79.2	96.4
(d) BP (observed)	TEST:	14.9***	71.6*	81.8	81.4***	96.7
ID3 Improvement: (c)-(a)	TEST:	3.4***	4.5***	2.8***	2.0**	0.3
BP Improvement: (d)-(b)	TEST:	0.2	0.9	0.7	0.0	0.1

sequence is evaluated and the results concatenated to obtain a bit string of length $k \cdot 26$. Then, this bit string is mapped to the nearest of the bit strings observed for this sequence in the training set (ties are broken in favor of the more-frequently occurring bit string). 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 $k-1$. We call this technique "block" decoding.

Table 8 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). Furthermore, the correlation coefficient between $X_{ID3block}$ and $X_{BPblock}$ is .6747, which is a substantial increase compared to .5508 for legal decoding. Hence, block decoding also makes the performance of ID3 and BP much more similar.

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 736 windows previously misclassified by "legal" decoding, it also loses 181 windows that were previously correctly classified by "legal" decoding. Similarly, there is only a .7746 correlation between $X_{BPlegal}$ and $X_{BPblock}$ (reflecting a gain of 433 and a loss of 226 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

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 917 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 917 windows. This would yield a performance of 79.9% at the letter level, which would be quite good.

Other directions for improving these algorithms include (a) design of better error-correcting codes, (b) block decoding using overlapping blocks, (c) direct analysis of the training set to identify morphemes.

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.

Table 8: 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:	14.7***	70.9***	81.1***	81.4***	96.6
(c) ID3 (block)	TEST:	17.5	73.2	83.8	80.4	96.7
(d) BP (block)	TEST:	19.9***	73.8	83.9	81.5*	96.7
ID3 Improvement: (c)-(a)	TEST:	7.9***	7.6***	5.1***	3.2***	0.6*
BP Improvement: (d)-(b)	TEST:	5.2***	2.9***	2.8***	0.1	0.1

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 it should be able to handle much wider windows without problems.

General solutions to the other two problems mentioned by Klatt appear to be quite challenging. Indeed, machine learning techniques have some distance to go before they match the performance of the best human-engineered rule sets. Klatt cites Bernstein and Pisoni (1980) as measuring the performance of their rule system to be 97% at the phoneme level. By comparison, ID3 trained on 19,003 words and tested using "block" decoding is 91.5% correct at the phoneme level (i.e., an error rate nearly three times as bad).

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 learning performance 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.

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 "block" decoding technique to ID3, the level of performance of the two algorithms in classifying test cases becomes statistically indistinguishable (at the letter and phoneme levels). Consequently, in tasks similar to the text-to-speech learning task, ID3 with block decoding is clearly the algorithm of choice—particularly for initial exploratory studies, where its speed is a tremendous advantage.

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. 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

Bernstein, J., and Pisoni, D. B., (1980). Unlimited text-to-speech system: description and evaluation of a microprocessor-based device. *Proceedings of the Int. conf. Acoust. Speech Signal Process, ICASSP-80*, 576–579.

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

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.