Learning New Articulator Trajectories for a Speech Production Model using Artificial Neural Networks

C. S. Blackburn and S. J. Young
Cambridge University Engineering Department (CUED), England
email: csb@eng.cam.ac.uk

ABSTRACT

We present a novel method for generating additional pseudo-articulator trajectories suitable for use within the framework of a stochastically trained speech production system recently developed at CUED. The system is initialised by inverting a codebook of (articulator, spectral vector) pairs, and the target positions for a set of pseudo-articulators and the mapping from these to speech spectral vectors are then jointly optimised using linearised Kalman filtering and an assembly of neural networks. A separate network is then used to hypothesise a new articulator trajectory as a function of the existing articulators and the output error of the system. The techniques used to initialise and train the system are described, and preliminary results for the generation of new pseudo-articulatory inputs are presented.

1. Introduction

Articulatory speech synthesis from text requires the specification of a set of articulator trajectories corresponding to a time-aligned phoneme string, together with a mapping from these trajectories to output speech. This mapping is frequently an explicit model of the human vocal tract [6, 8, 10], which theoretically provides the ability to produce very high quality speech waveforms incorporating time-domain modelling of co-articulation. In practice however, the performance of such systems is limited by model inaccuracies and in this paper we propose an alternative system in which a stochastically-trained model learns the mapping from articulatory to acoustic space [1].

We therefore relax the constraint that the system exactly mimic human physiology and instead use a set of “pseudo-articulators” [7] which fulfil roles similar to those of human articulators but whose positions are iteratively re-estimated from the training data. Initial articulator trajectory specification is achieved using an “inverse” model to map parametrised speech into articulator positions or vocal tract areas. We use a Kelly-Lochbaum synthesiser [5, 8] to generate a codebook of (articulator vector, spectral vector) pairs [9] which we invert using dynamic programming (DP) incorporating both acoustic and geometrical constraints on the articulator trajectories.

Target positions for the pseudo-articulators for each phoneme are estimated from the initial trajectories obtained from the DP algorithm and are used to re-construct trajectories corresponding to the training speech, incorporating an explicit model of co-articulation. These target positions are then iteratively re-estimated using linearised Kalman filtering and an assembly of neural networks which map from articulator positions to output speech.

Since the system is not constrained to the use of physiologically plausible articulators, it is possible to improve modelling accuracy by adding new articulators during the training process. We use a novel extension of the back-propagation algorithm to allow an artificial neural network to learn a new *input signal*, which when combined with the original pseudo-articulator inputs provides a significant reduction in training error. While several architectures have previously been proposed for the addition of hidden layer units to a network [4], the generation of a new input signal in this way appears to be novel. A brief overview of the basic speech production system will be given before providing the details of the generation of new articulators.

2. Speech production system

Five pseudo-articulators as used in [7] were sampled at regular intervals and used to determine a set of vocal tract area functions suitable for use in a Kelly-Lochbaum synthesiser which incorporates a transmission loss model and separate oral and nasal tracts. A sampling frequency of 16kHz was used and in all 102488 speech waveforms were generated, each of which was parametrised as a 12-dimensional liftered cepstral vector to give a codebook of 102488 (articulator vector, spectral vector) pairs.

A training speech database comprising 600 sentences of one adult male from the speaker-dependent training portion of the Defence Advanced Research Projects Agency Resource Management corpus was also coded into 12-dimensional cepstral vectors, and dynamic programming was used to find the best pseudo-articulator trajectory corresponding to each vector sequence. The cost function used incorporates both the acoustical mismatch between the parametrised training speech vector and the codebook acoustic vectors *and* the geometrical mismatch between successive articulatory vectors. To reduce the computational load, a sub-optimal search was used in which only the 500 codebook vectors with the best acoustic match were considered at each step.

The result of this process is a set of pseudo-articulator trajectories corresponding to the parametrised training speech vector sequences. Statistics describing the observed position of each of the pseudo-articulators during the production of each phoneme are determined by sampling the values of the pseudo-articulator trajectories at the midpoint of each occurrence of each phoneme to give initial estimates of target mean positions and covariance matrices $\hat{P}$. Although the word "target" is used here, we are in fact measuring the *achieved* position of each pseudo-articulator at the phonemic midpoints; the underlying target towards which an articulator was heading may never be reached in practice.

The pseudo-articulator trajectory corresponding to any arbitrary time-aligned phoneme string can then be determined by applying an explicit co-articulation model to the phonemic target means and using piece-wise linear interpolation constrained to pass through the average of two adjacent target means at the phonemic boundary [1, 2].

2.1. System training

The system is trained using the following iterative re-estimation process:

Repeat:

- Train a separate neural network to approximate the function from the pseudo-articulator trajectories of each phoneme to the output speech.
- Re-estimate the position of each pseudo-articulator at the phonemic midpoints using the linearised Jacobian matrices of the networks and linearised Kalman filtering.
- Compute the statistics of the new articulator positions for each phoneme and generate new articulator trajectories corresponding to the training speech from these new statistics.

The performance and architecture of the networks used are not crucial to the training process since their purpose is only to approximate the function from articulatory to acoustic space so that the linearised Jacobian matrix H can be used to re-estimate the phonemic targets; once the re-estimation is completed however, their performance is optimised as far as possible.

We trained feed-forward multi-layer perceptrons (MLPs) with 5 inputs, 30 hidden units, 24 outputs and sigmoid non-linearities at the hidden units using resilient back-propagation (RPROP) for 1000 batch update epochs, giving mean errors in estimated spectral coefficients of around 10%. The training set output vectors were 24-dimensional mel-scaled log spectral coefficients.

The global error covariance matrix R for each network mapping is estimated from its performance on an unseen test set, and the Jacobian matrix H is found by extending the usual error back-propagation formulae to evaluate the derivative of each output with respect to each input:

$$\frac{\partial y_k}{\partial y_i} = \sum_j (w_{ij} w_{jk} y_j (1 - y_j))$$

where y_i, y_j, y_k are the outputs of nodes in the input, hidden and output layers respectively and w_{ij}, w_{jk} are the input-hidden and hidden-output weights respectively. If the initial estimate of a phoneme's articulatory target mean vector is denoted $\hat{\mathbf{x}}$, with associated covariance matrix $\hat{P}$ and corresponding parametrised speech vector $\mathbf{z}$, and if the neural mapping is denoted $h()$ with Jacobian matrix H at the target estimate and output error covariance matrix R, the target vector can be re-estimated using linearised Kalman filtering as:

$$\mathbf{x} = \hat{\mathbf{x}} + \hat{P} H^T (H \hat{P} H^T + R)^{-1} (\mathbf{z} - h(\hat{\mathbf{x}}))$$

This gives a re-estimated target vector for each occurrence of each phoneme, from which new target mean and covariance statistics are computed. Updated pseudo-articulator trajectories are then derived and the networks retrained. This process is iterated to obtain an optimum set of phoneme targets from which speech is synthesised.

3. Generation of new inputs

In the speech production model described above, a partitioning of the input space into a discrete set of sub-spaces corresponding to 47 different phoneme classes is known *a priori*, allowing us to divide the problem of determining the mapping from articulator space to acoustic space into 47 sub-tasks, each of which is approximated by a separate neural network. We shall show that this knowledge of a partitioning of the input space can also be exploited to generate new input trajectories for the networks which lead to an overall increase in model accuracy.

If a neural network is trained using mean squared error (MSE) as a cost function to approximate a mapping from smooth functions at its inputs to smooth functions at its outputs, we expect the error at each output to be roughly zero mean over the entire training set. We trained a single network to approximate the mapping from pseudo-articulator trajectories to output speech vectors for all phonemes, and a typical plot of the output error signals during a single sentence is shown in Figure 1, where phonemic boundaries are marked as vertical lines.

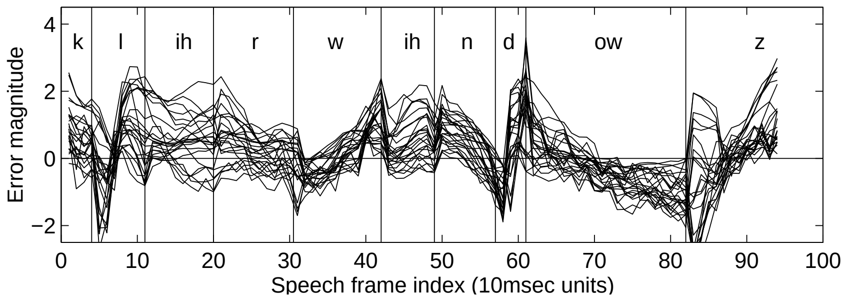

Figure 1: Variation in error at each of 24 network outputs over the course of the sentence "clear windows".

The mean error for each output over the course of the sentence is approximately zero, however within each phoneme there are clearly systematic variations in the error signal. It should therefore be possible to derive a new input for the network which is correlated with these systematic variations, in which case we would expect the overall network training error to decrease if we re-train the network on a data set augmented by this input.

By examining many error plots such as the above we find that different instances of the same phoneme have similar error signals – for example the the two occurrences of /ih/ in Figure 1 – which are in general not constant over the duration of a phoneme, but follow trajectories which are affected by the *context* in which the phoneme occurs. Therefore, while some reduction in the error magnitude could be achieved either by subtracting the mean error for each phoneme from the output signal according to the phonemic class of the current input[1], or by providing an additional input which identifies this phonemic class, a preferable solution would be to generate a new trajectory which incorporates this contextual variation.

If a suitable set of means for such an input were determined for each phoneme, a context-sensitive trajectory could be defined using piece-wise linear interpolation as described above, allowing a new input trajectory to be generated for an arbitrary input phoneme string. We use a single neural network trained on all the speech data to learn this new input since to do so with 47 different networks would result in a highly discontinuous solution.

3.1. System architecture

The architecture shown in Figure 2 was used, in which a conventional feed-forward MLP represented by the solid nodes and connecting links is trained to approximate a mapping between the known inputs and the outputs. The parameters of this network (weights and biases) are then frozen, and the additional structure indicated by hollow node symbols and dashed lines is added. A number of new hidden nodes are provided, which are connected to both the original inputs and the single new input, as well as to the output nodes of the original network.

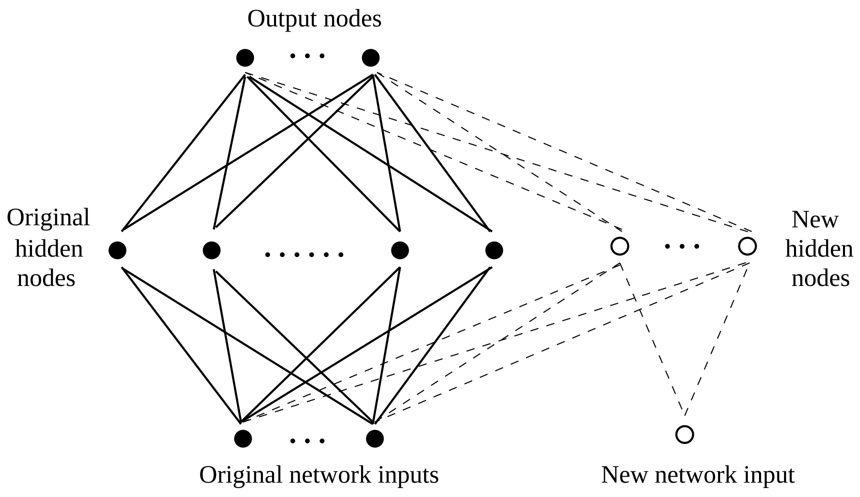

Figure 2: Network architecture for the generation of a new input.

The parameters of the new structure are then initialised to small random values, with the connections from the original inputs to the new hidden nodes setting an initial operating point in the weight space of the new hidden layer. The *error signal* at the output of the original (fixed) network is then back-propagated through the new network structure

[1] An effect implicitly achieved when using 47 separate networks.

to the new input values, which are initialised to zero for all training frames. The partial derivative of the error with respect to the input node value is derived in a similar way to the expression for the derivative of each output with respect to each input, and is:

$$\frac{\partial E}{\partial y_i} = \sum_j (w_{ij} y_j (1 - y_j) \sum_k (w_{jk} (y_k - t_k)))$$

where y_i, y_j, y_k are the outputs of nodes in the input, hidden and output layers respectively, w_{ij}, w_{jk} are the input-hidden and hidden-output weights respectively, E is the sum squared error at the outputs and t_k is the target output. The new inputs are then updated using:

$$y_i = -\eta * \frac{\partial E}{\partial y_i}$$

where η is analogous to the learning rate used in standard error back-propagation. Once a new input value has been computed in this way for every training frame, the parameters of the new network structure are optimised to produce a signal at the output nodes which approximates the negative of the original error E. After some number of iterations of this optimisation, the new (reduced) output error is once again back-propagated through to the new network inputs, which are updated once again. This process, similar to the Expectation-Maximisation (EM) algorithm [3], is continued until an optimum set of new input values has been determined. Due to the noisy nature of most output error signals, this new input will itself in general be noisy, so that some smoothing is required before extracting its systematic characteristics for use in generating the new input signal for new data.

Unlike standard back-propagation, this technique is not sensitive to changes in the value of η – which simply affects the magnitude of the new input signal – but *is* sensitive to the number of epochs of parameter optimisation performed per input update. After each update of the input values, the parameters of the new network structure are optimised to reduce the output error. If this optimisation is allowed to continue for a large number of iterations the parameters become highly tuned to the current input values, so that when the input values are next updated there is likely to be a large mis-match between these and the highly optimised parameters.

The solution to this problem for difficult learning tasks is to use only a small number of optimisation epochs per input update so that the new input values and the parameters of the new network structure jointly converge to a solution in a smooth sense, a process analogous to generalised EM. The new input learned will in general be a non-linear function of both the output signals of the original network and the original network inputs.

3.2. Example for an artificial system

To investigate the properties of the algorithm just described, an artificial data set was generated by taking two non-linear combinations of three basic functions:

$$x_1 = t^2, \quad x_2 = t + 1, \quad x_3 = 0.2sin(10t)$$

for values of t ranging from -1 to 1 in steps of size 0.01; the two non-linear functions used were:

$$f_1 = x_1 x_2 + x_3, \quad f_2 = 3 - x_1 - 0.5x_2^2 - 0.5e^{2t} x_3$$

to which we added zero-mean white noise of maximum absolute value 0.2. An MLP with 2 inputs, 4 hidden nodes and two outputs was trained using RPROP for 1000 batch update epochs on a data set consisting of the two inputs x_1 and x_2 and the noisy outputs f_1 and f_2. The input x_3 was chosen to be a sinusoid to produce systematic variations in the error signal, and was not supplied to the network. Two outputs functions were used since training with a single output results in the trivial solution of the error signal being reproduced as the new input.

The technique described above was used to generate a new input for the network by using 3 new hidden nodes, a learning rate of 1.0 and training for 50 input re-estimation iterations, each of which incorporated 100 epochs of parameter optimisation for the new network. The noisy new input signal generated was then smoothed using a third order butterworth low-pass filter with cutoff frequency one tenth of the Nyquist frequency. A separate neural network was then re-trained using the two original inputs x_1 and x_2 *and* the new input, again with 4 hidden nodes and the two output targets f_1 and f_2.

Due to its extra input node, the latter network has more parameters than the original system: 3 inputs, 4 hidden nodes and 2 outputs gives 26 parameters, whereas (2,4,2) gives only 22. Hence we trained an additional network for comparison on the two inputs x_1 and x_2, this time with 5 hidden nodes, where a (2,5,2) structure gives 27 parameters which is one more than the network with 3 inputs. Table 1 shows the results for the two networks, where the network incorporating the new input has resulted in a reduction in MSE of approximately 56%.

These results are shown graphically in Figure 3. The first sub-figure shows the target output functions before and after adding noise, while the second shows both noisy and smoothed generated network inputs together with the

Network structure	Number of parameters	MSE	improvement (%)
(2,5,2)	27	0.0119	-
(3,4,2)	26	0.00526	55.8

Table 1: Performance of networks on artificial data.

original missing input (dashed), where a strong correlation can be seen between the two. The final sub-figure shows the target and actual network outputs. The dotted plot is the noisy target output, while the dashed plot corresponds to the original 2-input network, and the solid plot to the new 3-input network. Clearly the 3-input network has learned a greatly superior mapping despite having one less parameter.

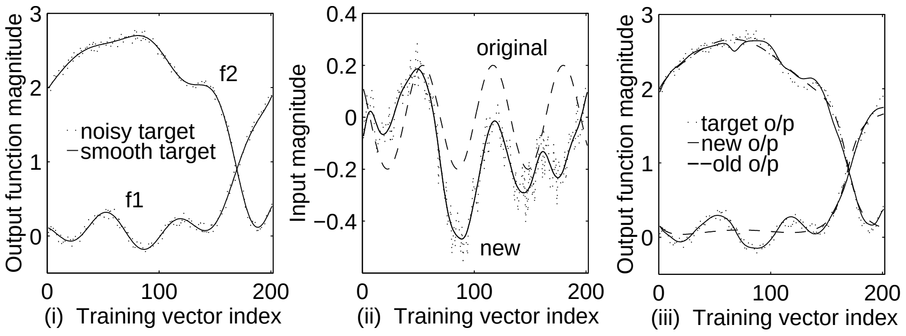

Figure 3: (i) Two network target outputs, both before and after adding zero mean white noise (ii) The new (learned) network input signal both before (dotted) and after (solid) smoothing, and original missing input x_3 (dashed) (iii) Noisy network target outputs, and outputs of both the original network and that trained with the new input.

3.3. Application to speech production

In applying the input generation technique to our speech production system, we need to show not only that it is possible to generate a new input which results in a reduction in the training error as in the previous example, but also that this new input can be characterised in a general way such that it can be generated for any arbitrary vector of ordinary network inputs, given the information as to the input phoneme class at any point in time.

An MLP with 5 inputs, 30 hidden nodes and 24 outputs was trained, once again using 1000 batch update epochs of the RPROP algorithm, to learn the mapping from co-articulated pseudo-articulator trajectories to speech spectral vectors for 400 sentences of training data comprising a total of 122102 vectors. We then added 10 new hidden nodes and trained the new network structure to learn a new input for this system. During this stage of the training the constant η was set to 1.0 and 100 re-estimation iterations were performed, with 1 epoch of RPROP optimisation of the new network parameters for each update of the input values.

This minimal amount of parameter optimisation per iteration was necessary to ensure smooth convergence to an optimal set of input values, and the MSE decreased during training from 1.5007 to 0.5497. The new input trajectory obtained was smoothed in individual sections corresponding to input training sentences using a third order butterworth low-pass filter with cutoff frequency one fifth of the Nyquist frequency, and then sampled at the phoneme midpoints to obtain statistics for the mean position of the new input for each phoneme.

New input trajectories were constructed from these means using the same piece-wise continuous interpolation used for the original inputs, and the new trajectory so formed was added to the original data set. A network with 6 inputs, 31 hidden nodes and 24 outputs was then trained using 1000 batch update epochs of the RPROP algorithm, to learn the mapping from the augmented input set to the original outputs. This (6,31,24) network has 985 parameters, so to ensure a fair comparison a separate network with 32 hidden nodes was trained on the original input set, giving a (5,32,24) structure comprising 984 parameters.

Network structure	Number of parameters	MSE	improvement (%)
(5,32,24)	984	1.524	-
(6,31,24)	985	1.275	16.3

Table 2: Performance of networks on speech data.

The results are given in Table 2, where we see that with a comparable number of parameters the system which uses the new input has 16.3% less MSE than the system trained on the original inputs. We emphasise that the new input trajectory used was *not* that learned directly by the augmented network structure, but was generated from the statistics of this trajectory. Hence a new input trajectory such as this can be generated for an arbitrary input phoneme sequence. Figure 4 shows the MSE for both the (5,32,24) network trained on the original data and the

(6,31,24) network trained on the augmented data set. The plots exclude the first 10 epochs of training to provide reasonable scaling in the y-axis.

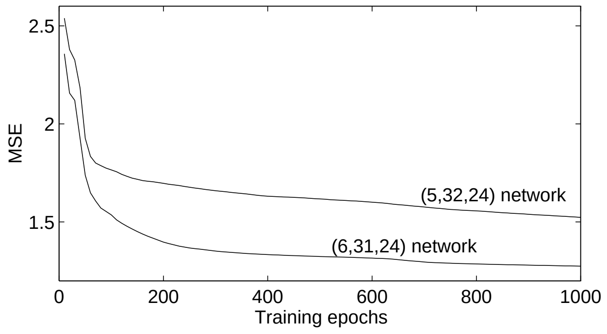

Figure 4: Network error curves for original and augmented data sets.

4. Conclusions

This paper has presented a novel technique for generating a new input for an artificial neural network which has been trained to learn the mapping from a set of smooth input functions to a corresponding set of smooth output functions, under the condition that a subdivision of the input space into distinct classes is known a priori at each time step. If the error of such a system shows systematic variations which are correlated with changes in the input class and dependent upon the input context, statistics describing the form of the new input can be computed which allow such an input to be generated given any set of original trajectories.

The technique has been demonstrated on both an artificial system and in the context of a pseudo-articulatory speech production model recently developed at CUED, and in both cases was seen to provide a significant reduction in output error.

Other fields where this technique may have applications include slowly parameter-varying control systems, in which an interpolation is performed between a number of linear models which approximate a non-linear mapping. If the output error of the system has systematic variations which are correlated with the particular linear model being used, a new signal could be derived as a function of the output error and the slowly-varying parameter so as to reduce the overall system error.

This system is still under development, and many questions have yet to be resolved. The convergence and stability criteria of the re-estimation technique for generating new inputs need to be investigated, as does the sensitivity of the system to the initialisation conditions. The viability of the model as applied to our speech production system seems excellent however, and the initial results obtained are extremely encouraging.

5. References

[1] C. S. Blackburn and S. J. Young. “A novel self-organising speech production system using pseudo-articulators”. *Int. Congr. Phon. Sc.*, 1995. Accepted for publication.

[2] C. S. Blackburn and S. J. Young. “Towards improved speech recognition using a speech production model”. *Europ. Conf. Sp. Comm. Tech.*, 1995. Accepted for publication.

[3] A. P. Dempster, N. M. Laird, and D. B. Rubin. “Maximum likelihood from incomplete data via the EM algorithm”. *J. Roy. Stat. Soc.*, 39(B1):1–38, 1977.

[4] S. E. Fahlman and C. Lebiere. “The Cascade-Correlation learning architecture”. Technical Report CMU-CS-90-100, Carnegie Mellon University, 1990.

[5] J. L. Kelly Jr. and C. Lochbaum. “Speech synthesis”. In *Sp. Comm. Sem.*, Stockholm, 1962.

[6] P. Mermelstein. “Articulatory model for the study of speech production”. *J. Acoust. Soc. Am.*, 53(4):1070–1082, 1973.

[7] P. Meyer, R. Wilhelms, and H. W. Strube. “A quasiarticulatory speech synthesizer for German language running in real time”. *J. Acoust. Soc. Am.*, 86(2):523–539, 1989.

[8] P. Rubin, T. Baer, and P. Mermelstein. “An articulatory synthesizer for perceptual research”. *J. Acoust. Soc. Am.*, 70(2):321–328, Aug. 1981.

[9] J. Schroeter and M. M. Sondhi. “Techniques for Estimating Vocal-Tract Shapes from the Speech Signal”. *IEEE Trans. Sp. Aud. Proc.*, 2(1):133–150, Jan. 1994.

[10] M. M. Sondhi and J. Schroeter. “A hybrid time-frequency domain articulatory speech synthesizer”. *IEEE Trans. Acoust. Sp. Sig. Proc.*, ASSP-35(7):955–967, July 1987.