17

# Comparison of feed-forward and recurrent sensitivities in speech recognition

**GARY M. KUHN and RAYMOND L. WATROUS**
*Siemens Corporate Research, Princeton, NJ, USA*

## 17.1 Introduction

In earlier work, we defined and calculated the sensitivity of each output unit of a feed-forward network to each input feature in its training set at each point in time. This calculation suggested the need for a change in architecture, and led to a subsequent **joint** optimization of the network and its input features.

Now we define and calculate the sensitivity of a **recurrent** network to each input feature in the same training set at the same points in time. This calculation makes it possible to quantify the extent to which the two types of networks have similar sensitivities to their inputs.

## 17.2 The networks

In Kuhn and Herzberg (1991) our feed-forward network was a 'time-delay neural network' with one hidden layer:

$$y_k(t) = S\left( w_{\theta k} + \sum_{j=1}^{9} \sum_{l=}^{1,3,5} w_{jkl} S\left[ w_{\theta j} + \sum_{i=1}^{17} \sum_{d=1}^{4} w_{ijd} y_i(t-l-d) \right] \right),$$

where $y_k(t)$ is the output for class $k$ at time $t$, $S$ is the symmetric sigmoid $y = 2(1 + e^{-x})^{-1} - 1$, $S(\cdot) = y_k(t)$, $S[\cdot] = y_j(t - l)$ is the output of hidden unit $j$ at time $t - l$, $y_i\,(t - l - d)$ is the output of input unit $i$ at time $t - l - d$, $l$ is a time-**lag** from hidden to output unit, $d$ is a time-**delay** from input to hidden unit, and $w_{\theta j}$ and $w_{\theta k}$ are bias values.

Appeared in *Artificial Neural Networks for Speech and Vision.* Edited by Richard 1. Mammone. Published in 1993 by Chapman & Hall, London. ISBN 0 412 54850 X

The recurrent network was identical to the feed-forward network except that it also had output self-loops with unit time delay:

$$y_k(t) = S\left( w_{kk} y_k(t-1) + w_{\theta k} + \sum_{j,l} w_{jkl} S\left[ w_{\theta j} + \sum_{i,d} w_{ijd} y_i(t-l-d) \right] \right).$$

The feed-forward network was trained to discriminate the spoken letter names 'b', 'd', 'e' and 'v', and achieved 88.5% accuracy on test examples. The recurrent network was trained on the same task and achieved 89.6% accuracy on the same test examples (Kuhn and Herzberg, 1991).

## 17.3 Feed-forward sensitivity

Kuhn (1992) defined and calculated the sensitivity of each output unit of the feed-forward network to each input feature observed in the network's training set at each point in time. This sensitivity was time-dependent, to avoid integrating over times in the input when sensitivities might have opposite sign.

The feed-forward sensitivity $s_i^k(t)$ of output unit $k$ to input $i$ at time $t$ was defined as

$$s_i^k(t) = \sum_{\tau} \frac{\partial y_k(t+\tau)}{\partial y_i(t)}$$

where

$$\frac{\partial y_k(t+\tau)}{\partial y_i(t)} = \frac{\partial y_k(t+\tau)}{\partial x_k(t+\tau)} \sum_{l+d=\tau} \frac{\partial x_k(t+l+d)}{\partial y_i(t)}$$

and

$$\frac{\partial x_k(t+l+d)}{\partial y_i(t)} = \sum_j w_{jkl} \frac{\partial y_j(t+d)}{\partial x_j(t+d)} w_{ijd}$$

yielding

$$s_i^k(t) = \sum_{\tau} \frac{\partial y_k(t+\tau)}{\partial x_k(t+\tau)} \sum_{l+d=\tau} \sum_j w_{jkl} \frac{\partial y_j(t+d)}{\partial x_j(t+d)} w_{ijd}.$$

It is easy to confuse sensitivity $s_i^k(t)$ with the derivative of the output of unit $k$ at time $t$ with respect to feature $i$. Note that the derivative requires a sum over those earlier times whose feature $i$ **affects** output $k$ at time $t$, while the sensitivity requires a sum over all later times $t + \tau$ at which output $k$ is **affected by** feature $i$ at time $t$.

The sensitivity is also similar to the gradient of the system error at unit $k$ with respect to the weight from feature $i$. But the sensitivity omits the first factor of the error gradient, namely the derivative of the error with respect to unit $k$'s output, and it substitutes $w_{ijd}$ for the last factor of the error gradient, namely for $y_i(t-l-d)$.

To compare sensitivities when the feed-forward network was responding as desired with those when it was not, we ordered the training examples by how well they were discriminated (Kuhn, 1992). At the well-discriminated end of the ordered list of training examples, we found that there was little residual sensitivity: the network responses were locally robust to perturbations of the inputs, and the responses were correct.

At the poorly-discriminated end of the ordered list of training examples the network actually mis-classified the last 10% of the training examples. Here we could also have found little residual sensitivity, which would have meant that the network responses were once again locally robust to perturbations of the inputs, but this time, the responses would have been wrong. Instead, what we found was (1) there was much more residual sensitivity for the poorly discriminated training examples, (2) some input features had a net negative bias of their sensitivity while others had a net positive bias, and (3) the total absolute sensitivity, i.e. the magnitude of the sensitivity, differed markedly from feature to feature.

The residual sensitivity for the poorly-discriminated training examples was both bad news and good news. It was bad news because we always wanted robust, state-to-state jumps in the outputs of our network, but we sometimes got volatile and weak ramping of the outputs. On the other hand, the residual sensitivity was also good news, because it indicated that there was still more slope to be exploited by the training algorithm.

## 17.4 Feature optimizing

The biases and magnitudes of the sensitivities suggested that a global one-to-one affine transformation should be applied independently to each feature. The recognizer was augmented to implement this transformation, by adding what we can think of as a new optimizing layer with a 1-element localized receptive field. Figure 17.1 shows both the original and the augmented recognizer.

Writing the original inputs-to-hidden transformation as:

$$y_j(t) = S\left[ w_{\theta j} + \sum_{i,d} w_{ijd}(0 + 1 y_i(t-d)) \right]$$

the augmented transformation becomes:

$$y_j(t) = S\left[ \hat{w}_{\theta j} + \sum_{i,d} \hat{w}_{ijd}(w_{\theta i} + w_{ii} y_i(t-d)) \right].$$

We can set the linear part of the original transformation equal to the linear part of the augmented transformation

$$w_{\theta j} + \sum_{i,d} w_{ijd} y_i(t-d) = \hat{w}_{\theta j} + \sum_{i,d} \hat{w}_{ijd}(w_{\theta i} + w_{ii} y_i(t-d))$$

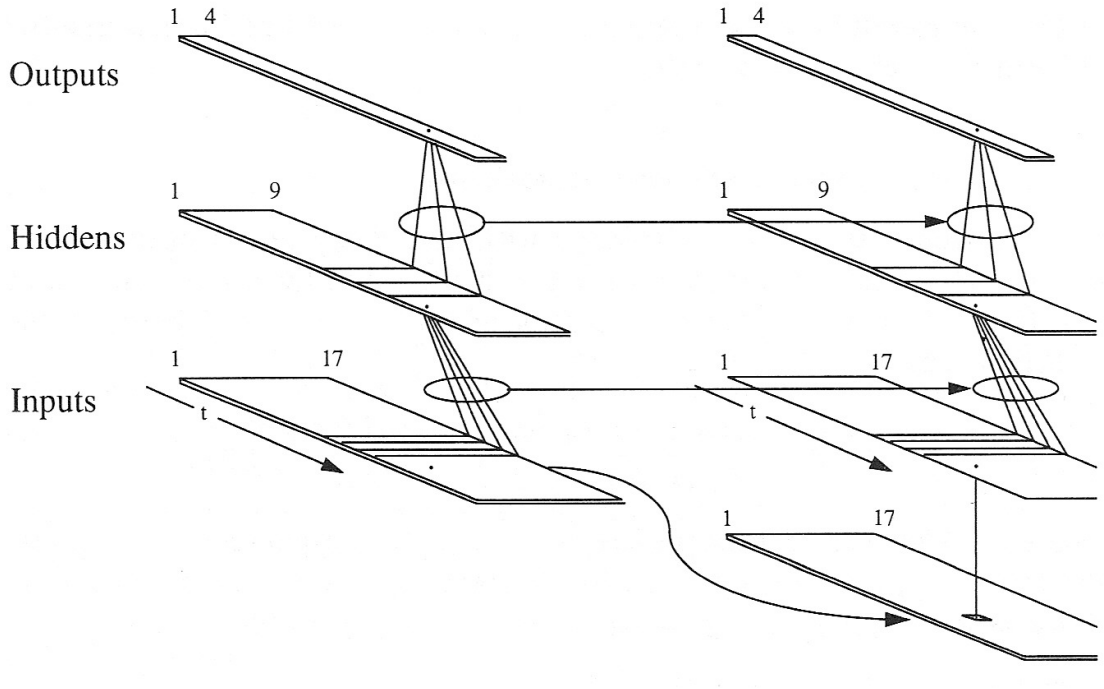


Figure 17.1 *Original and augmented recognizer.*

and then set the original weights $w_{\theta j}$ and $w_{ijd}$ equal to the corresponding quantities from the augmented system

$$w_{\theta j} = \hat{w}_{\theta j} + \sum_{i,d} \hat{w}_{ijd} w_{\theta i}, \quad w_{ijd} = \hat{w}_{ijd} w_{ii}.$$

Does this equivalence mean that the 1-to-1 affine transformation of the inputs was useless? It does not. Any number of pairs of multipliers can produce the same (dot) product, but not all pairs will train the first member of the pair (the weight) equally easily, because training is proportional to the second member of the pair. By allowing global changes to individual components in the input, additional **paths** through weight space can open up to perhaps previously unreachable solutions. This can improve the trainability of the network, as well as the interpretability of the features. It is true however, that after training we could fold the feature optimizing weights back into the input-to-hidden weights of an architecture like that of the original classifier.

With the added layers, the network can concentrate the scale and offset of an input feature at a single location, via the 1-to-1 affine transformation, rather than having to distribute that scale and offset across all the connections and units that follow an input unit. For example, if weight decay is imposed on our optimized-to-hidden and hidden-to-output weights, more of the input scaling can be forced into the 1-to-1 affine transformation.

As it turned out, further joint optimization of the augmented feed-forward network did increase discrimination of the test examples to 89.6%, while further optimization of the

original network for the same additional number of iterations yielded no increase.

## 17.5 Forward calculation of recurrent sensitivity

The recurrent network has self-loops with unit delay on its output units. Because of this added recursive structure, a change in the input to unit $k$ at any time will affect all further outputs of that same unit. Therefore the recurrent sensitivity is

$$s_i^k(t) = \sum_{\tau} \frac{\partial y_k(t+\tau, \ldots, T)}{\partial x_k(t+\tau)} \sum_{l+d=\tau} \sum_{j} w_{jkl} \frac{\partial y_j(t+d)}{\partial x_j(t+d)} w_{ijd}$$

where the difference between the feed-forward and recurrent sensitivities is that the factor $(\partial y_k(t+\tau,))/(\partial x_k(t+\tau))$ has become $(\partial y_k(t+\tau, \ldots, T))/(\partial x_k(t+\tau))$. Since we have $y_k(t+\tau, \ldots, T) = y_k(t+\tau) + \ldots + y_k(T)$, it follows that

$$\begin{aligned}
\frac{\partial y_k(t+\tau, \ldots, T)}{\partial x_k(t+\tau)} &= \frac{\partial y_k(t+\tau)}{\partial x_k(t+\tau)} \frac{\partial}{\partial y_k(t+\tau)} (y_k(t+\tau) + \cdots + y_k(T)) \\
&= \frac{\partial y_k(t+\tau)}{\partial x_k(t+\tau)} \left( 1 + \frac{\partial y_k(t+\tau+1)}{\partial y_k(t+\tau)} + \cdots + \prod_{n=t+\tau}^{T-1} \frac{\partial y_k(n+1)}{\partial y_k(n)} \right) \\
&= \frac{\partial y_k(t+\tau)}{\partial x_k(t+\tau)} \left( 1 + \sum_{m=1}^{T-t-\tau} w_{kk}^m \prod_{n=t+\tau}^{t+\tau+m-1} \frac{\partial y_k(n+1)}{\partial x_k(n+1)} \right),
\end{aligned}$$

where the first term is the only term that exists in the feed-forward case.

## 17.6 Backward calculation of recurrent sensitivity

The forward calculation of recurrent sensitivity can either be carried out in its entirety, or it can be truncated, or it can be ignored in favor of the following **backward** calculation of recurrent sensitivity.

We use the familiar chain rule to expand:

$$\frac{\partial y_k(t+\tau)}{\partial y_i(t)} = \frac{\partial y_k(t+\tau)}{\partial x_k(t+\tau)} \frac{\partial x_k(t+\tau)}{\partial y_i(t)}$$

and

$$\frac{\partial x_k(t+\tau)}{\partial y_i(t)} = \sum_{m,\beta} w_{mk\beta} \frac{\partial y_m(t+\tau-\beta)}{\partial y_i(t)}.$$

Thus

$$s_i^k(t) = \sum_{\tau \geq 0} \frac{\partial y_k(t+\tau)}{\partial x_k(t+\tau)} \sum_{m,\beta} w_{mk\beta} \frac{\partial y_m(t+\tau-\beta)}{\partial y_i(t)}.$$

If we compare with the gradient of a mean square error objective function:

$$\phi(w) = \frac{1}{2} \sum_{n=1}^{T} \sum_{k=1}^{O} (\mathrm{targ}_k(n) - y_k(n))^2$$

where

$$\nabla\phi(w) = \sum_{n=1}^{T} \sum_{k=1}^{O} (\mathrm{targ}_k(n) - y_k(n)) \frac{\partial y_k(n)}{\partial x_k(n)} \sum_{m,\beta} w_{mk\beta} \frac{\partial y_m(n-\beta)}{\partial w_{ij\lambda}},$$

we see that the computation of the output-input sensitivities can be organized very similarly to the back-propagation-in-time algorithm. That algorithm may be modified by injecting an error of 1 only for the selected output unit at each time step, and by extending the back-propagation one more step to compute the partials with respect to the inputs; the sensitivities may be read off directly from the backpropagation data structure.

## 17.7 Simulations

To apply these definitions, we made two new training runs on our speech data, and then calculated both the feed-forward and recurrent sensitivities in the backward direction, using the GRADSIM simulator (Watrous, 1992).

The speech data are divided into a training set of 672 utterances and a test set of 96 utterances. Both the feed-forward and the recurrent network were optimized using a conjugate gradient algorithm until the MSE was reduced to 0.008; at this point, discrimination of the training set was 90.0% for the feed-forward network and 91.5% for the recurrent network.

However, discrimination of the test set was only 78.1% for the feed-forward network and only 82.3% for the recurrent network. This is worse than the 88.5% and 89.6% previously reported on the same data (Kuhn and Herzberg, 1991). We point out that in the simulations reported here, as opposed to those reported by Kuhn and Herzberg (1991), the networks actually used the logistic non-linearity $y = (1 + e^{-x})^{-1}$ rather than the symmetric sigmoid, and the training runs did not include either a perodic cross-validation step or an alternation between global stochastic search and local gradient search.

After training, the feedback weights $w_{kk}$ for the spoken letter names 'b', 'd', 'e' and 'v' were, respectively, 4.2, 2.9, 0.4 and 1.2, indicating that the recurrent network did weight previous time more heavily for the relatively dynamic voiced stop-consonants 'b' and 'd'.
Figure 17.2-5 show, respectively, feed-forward sensitivities on the best discriminated training examples, feed-forward sensitivities on poorly-discriminated training examples, recurrent sensitivities on the same best-discriminated training examples and recurrent sensitivities on the same poorly discriminated training examples. Training example discriminability was measured using the feed-forward network. The measure was the smallest error for an incorrect class divided by the error for the correct class.

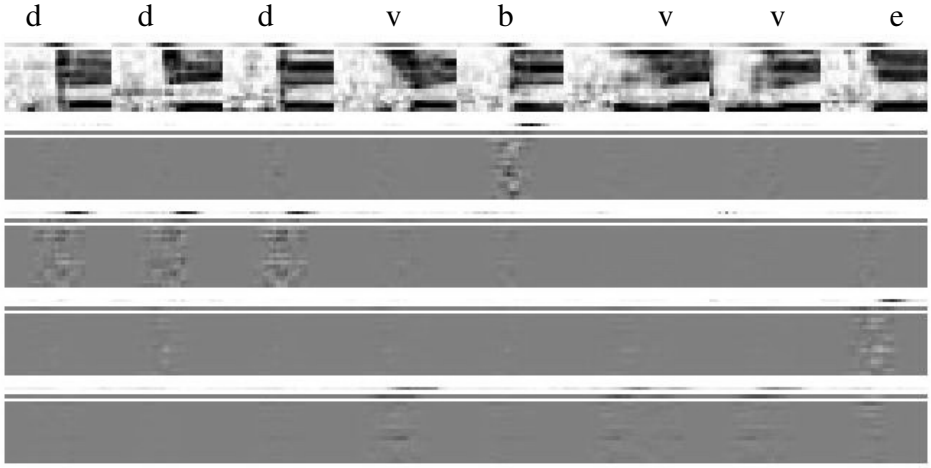


Figure 17.2 *Features and feed-forward sensitivities for best-discriminated training examples.*

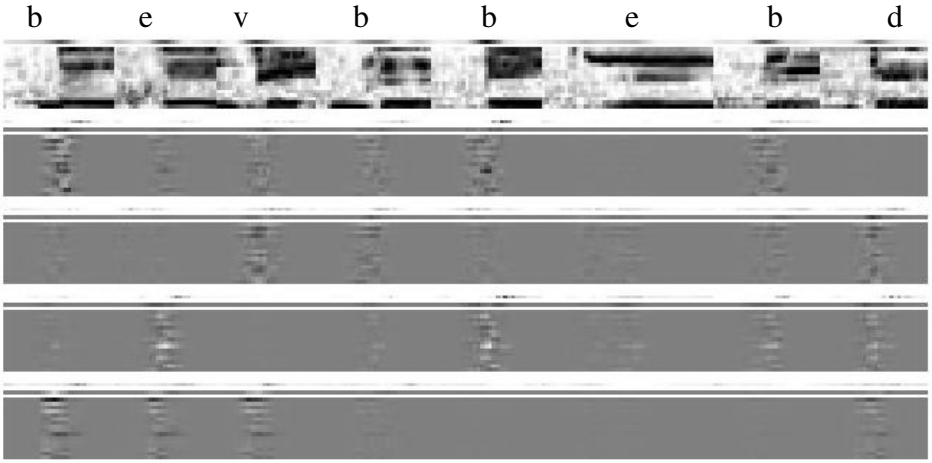


Figure 17.3 *Features and feed-forward sensitivities for poorly-discriminated training examples.*

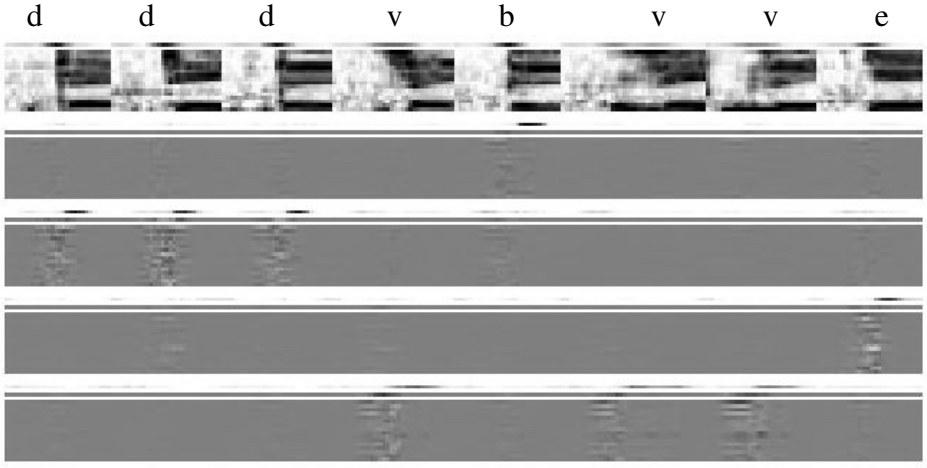


Figure 17.5 *Features and recurrent sensitivities for same training examples as in Figure 17.2.*

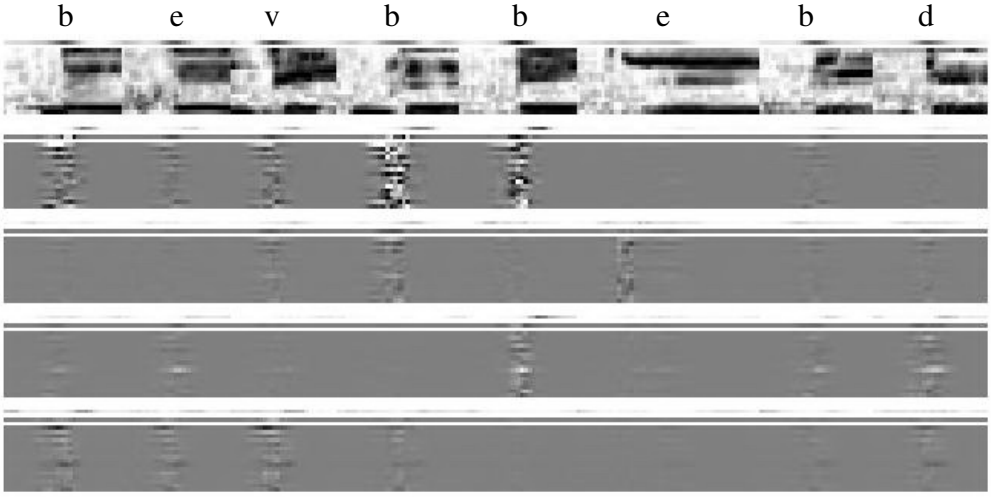


Figure 17.5 *Features and recurrent sensitivities for same training examples as in Figure 17.3.*

More specifically, Figure 17.2 shows the 8 best-discriminated utterances and Figure 17.3 shows the last four correctly discriminated and the first four mis-discriminated utterances, both for the feed-forward network. Figures 17.4 and 17.5 show the same well-discriminated and poorly-discriminated utterances, respectively, for the recurrent network.

There are five panels in each of the four figures. In the top panel there are 17 speech features as a function of time. From bottom to top, the features are 16 filter bank energies ordered from low frequency to high (Kuhn, 1992), and one event signal (Kuhn *et al.,* 1990) turning on and off above the filter energies. In panels 2-5, we see quantities relating to the 'b', 'd', 'e' and 'v' output units. These quantities, from bottom to top in each panel, are the sensitivity of the indicated unit to each of the filter bank energies, the sensitivity of the indicated unit to the event signal, and the response of the unit. The display of the sensitivities ranges from white, for large negative values, to medium grey, for zero values, to black, for large positive values.

In Figure 17.2, the feed-forward sensitivities for the well-discriminated examples, the overall sensitivity values are fairly uniform and close to zero. Slightly lighter or darker regions are evident in some areas, especially when the output unit corresponds to the token being uttered. The recurrent sensitivities for the well-discriminated utterances, in Figure 17.4, are in general quite similar to those of the feed-forward network. There is, however, less sensitivity of the recurrent b-unit for the 'b', and increased sensitivity of the recurrent v- and e-units in response to the final 'e'.

In contrast, there are many more extreme sensitivity values in the poorly-discriminated examples. Again, the feed-forward and recurrent sensitivities are similar. For example, both want a weaker second formant if the e-unit is to increase its response (incorrectly) to the third 'b'. Some recurrent sensitivities are however more salient, particularly those of the b-unit.

Note, finally, the alternation of the sign of the sensitivities across channels and panels at the same time. This alternation suggests that the network is trying to balance the residual sensitivities.

## 17.8 Discussion

We have touched on several topics that we now go over again, one by one.

The definition of the speech event signal is found in (Kuhn *et al.,* 1990), where the delayed event signal was first used as both the target signal during training, and as the hypothesized target signal during recognition.

The small improvement reported for the 1-to-1 affine transformation in Kuhn (1992) is similar to the small improvement reported for the 'autonorm layer' in Seidel *et al.* (1992), where the better the data was normalized prior to being input to the network, the less benefit there was to using the 1-to-1 affine transformation.

We looked for a revealing ordering of the sensitivity data and chose an ordering by discriminability. Other revealing orderings could be examined, such as those favored by projection pursuit (Friedman and Tukey, 1974).

The use of residuals in system identification, of which residual sensitivity is only one example, is discussed in generous detail by Draper and Smith (1981, Ch.3).

A thoughtful discussion that includes approaches to training large, fixed size networks by alternating between global random search and local gradient search, is found in Barron and Barron (1988).

We believe that the sensitivity analysis permitted us to find a better architecture for joint optimization **both** of the data representation **and** of the effective parameters of the classifier. This approach of joint optimization can be juxtaposed to an approach where one emphasizes the difference between optimizing data representations and optimizing the class of approximating functions, as in Geman *et al.* (1992).

## 17.9 Conclusion

To recapitulate, we have a feed-forward network and a recurrent network. Each has been trained to discriminate the speech sounds 'b', 'd', 'e' and 'v' using the same set of training data. The recurrent network performs slightly better on test data.

We have already defined and calculated the sensitivity of the outputs of the feed-forward network at each time to each input feature in the training set. This calculation suggested the need for a change in architecture and led to a subsequent **joint** optimization of the network and its input features.

We have now defined and calculated the sensitivity of the recurrent network at each time to each input feature in the training set. This calculation makes it possible to quantify the extent to which the two types of networks have similar sensitivities to their inputs.

**Acknowledgements**

We thank colleagues F. Block, R. Cohn, R. Mammone, C. Olano, V. Poor and M. Williamson for helpful discussions.

**References**

Barron, A.R. and Barron, R.L. (1988) Statistical Learning Networks: A Unifying View. *Computing Science and Statistics: Proc. 20th Symposium on the Interface,* pp. 192-203.
Draper, N.R. and Smith, H. (1981) *Applied Regression Analysis.* Wiley, New York, NY, pp.141-92.
Friedman, J. and Tukey, J. (1974) A projection pursuit algorithm for exploratory data analysis. *IEEE Trans. Computers,* 23,881-90.
Geman, S., Bienenstock, E. and Doursat, R. (1992) Neural networks and the bias/ variance dilemma. *Neural Computation,* 4,1-58.
Kuhn, G.M. (1992) Joint optimization of classifier and feature space in speech recognition. *IJCNN' 92*, IV, 709-14.
Kuhn, G.M. and Herzberg, N.P. (1991) Some variations on training of recurrent networks, in *Neural Networks: Theory and Applications,* (eds. R. Mammone and Y. Zeevi), Academic Press, New York, NJ, pp.233-44.
Kuhn, G.M., Watrous, R.L. and Ladendorf, B. (1990) Connected recognition with a recurrent network. *Speech Communication,* 9,41-9.
Seidel, F., Becks, K.H., Block, F. *et al.* (1992) B-Quark tagging using neural networks and comparison with a classical method. *Second Intl. Workshop on Software Engineering, Artificial Intelligence and Expert Systems for High Energy and Nuclear Physics,* La Londe-Les-Maures, France, January 13-18.
Watrous, R.L. (1992) GRADSIM: A connectionist network simulator using gradient optimization techniques. *Siemens Internal Report,* November 17.