Supervised Networks That Self-Organize Class Outputs

Ramesh R. Sarukkai
University of Rochester, Rochester, NY 14627 USA

Supervised, neural network, learning algorithms have proved very successful at solving a variety of learning problems; however, they suffer from a common problem of requiring explicit output labels. In this article, it is shown that pattern classification can be achieved, in a multilayered, feedforward, neural network, without requiring explicit output labels, by a process of supervised self-organization. The class projection is achieved by optimizing appropriate within-class uniformity and between-class discernibility criteria. The mapping function and the class labels are developed together iteratively using the derived self-organizing backpropagation algorithm. The ability of the self-organizing network to generalize on unseen data is also experimentally evaluated on real data sets and compares favorably with the traditional labeled supervision with neural networks. In addition, interesting features emerge out of the proposed self-organizing supervision, which are absent in conventional approaches.

1 Motivation

Supervised learning in conventional multilayered neural networks can be formulated without explicitly specifying output label functions. The goal of supervised learning systems is to determine a function $F(X)$, which "agrees" with the training examples of the form $(X_I, G(X_I))$. Usually the function $F()$ takes on values from a discrete set of "classes": $\{C_0, \ldots, C_K\}$. Many algorithms have been proposed for such a learning task: decision tree methods such as CART (Breiman et al. 1984) and multilayered feedforward networks (Rumelhart et al. 1987) trained by the "error-backpropagation" algorithm, to name just two. In practice, however, there is an intermediate mapping that is needed, namely, label assignments. Formally, distinct functions, $L_{C_I}$, are defined for each class $C_I$ and are referred to as the labels. Thus, the learning system finds an approximate definition of the function $F(X)$ such that, given training examples, $(X_I, G(X_I))$, $L(F(X_I)) = L(G(X_I))$.

The popular error-backpropagation algorithm (Rumelhart et al. 1987), for training multilayered, feedforward, artificial neural networks, has been formulated on the explicit assumption that an output labeling function $L()$ is available.

Neural Computation 9, 637–648 (1997) © 1997 Massachusetts Institute of Technology
Biologically, output labels are available for certain, but not all, input-output mappings. All of the above schemes share a common deficiency in that the labeling function, $L()$, has to be predetermined. Why not formulate self-organizing systems, which develop their own output codes based on the supervisory signals? This article presents an approach whereby conventional multilayered feedforward networks, under supervision, self-organize their own output labels. In particular, I show that supervised learning with neural networks can be viewed as a process of dimensionality reduction, with additional constraints: a within-class constraint specifying that the output labels (or projections to the output space) must be the same for inputs belonging to the same class and a between-class constraint specifying that the output labels should be different for different classes. Thus, the role of supervision is only to suggest to the learning machine which inputs belong together and which do not.

2 Separability Enhancement by Nonlinear Mappings

The idea of viewing supervised learning as a process of class separability enhancing, dimensionality reduction was first presented in Fisher’s (1936) classic work, which was the origin of discriminant analysis. Class-enhancing projections using nonlinear mappings have been extensively researched in classical feature extraction theory. Typically iterative techniques proceed by starting with random values and search for the “good” projections based on steepest-descent methods using criteria functions (Duda & Hart 1973; Fukunaga 1972; Young & Calvert 1974) such as monotonicity, stress, and continuity. The criteria functions are usually structure-preserving measures. The algorithms terminate when no further improvement can be obtained in the projections, and the mapping function is then usually found by a least-squares technique.

In the neural network literature (Rumelhart et al. 1987; Hertz et al. 1991), supervised learning is usually implicitly tied to the output labeling function. Perhaps this was due to the ability of elegantly formulating gradient weight adaptation equations (i.e., error is related to teaching output – actual output). Independent of my work, discriminant analysis neural networks have been studied (Kuhnel & Tavan 1991; Mao & Jain 1993); however, their formulation is based on principal component analysis (PCA) networks, and cannot be generalized to multilayered, feedforward networks. The learning vector quantization (LVQ) algorithms proposed by Kohonen (Hertz et al. 1991) can also be grouped with the class of discriminant analysis networks in discussion. Although supervision is used, the goal of LVQ is to find appropriate cluster centers in the input space rather than dimensionality reduction-enhancing class separability.

In this article, a different method of achieving supervised nonlinear projection is implemented with a conventional, multilayered feedforward network. Unlike existing iterative approaches to nonlinear class separability
Supervised Networks That Self-Organize Class Outputs

enhancement, the presented implementation of supervised projection using neural networks enables the simultaneous development of the whole nonlinear mapping function along with the class labels.

3 Supervised Projection Using Neural Networks

In this section, the learning algorithm that has been implemented is discussed briefly. For the sake of clarity, simple functions were chosen, although the basic idea can be extended to more complex and general functions.

Let $O_{pj}$ be the activation state of the $j$th output after the presentation of pattern $p$. Let $\Omega_k$ be the set of inputs belonging to class $k$. $E_{p_{1,j}}$ refers to the error with respect to pattern $p_1$ for output unit $j$.

The *within-class criterion* used is

\[
\forall \Omega_k, k = 1, \ldots, n \\
\forall p_1 \in \Omega_k.
\]

Minimize

\[
E_{p_{1,j}} = \frac{1}{2} \sum_{\forall p \in \Omega_k} [O_{p_{1,j}} - O_{p_{2,j}}]^2.
\]  
(3.1)

Note that the within-class criterion is minimized when output responses for patterns belonging to the same class are identical—that is, when $O_{p_{1,j}} = O_{p_{2,j}}$.

The *between-class criterion* is defined to be

\[
\forall \Omega_k, k = 1, \ldots, n \\
\forall p_1 \in \Omega_k.
\]

Minimize

\[
E_{p_{1,j}} = \sum_{\forall p \notin \Omega_k} 1 - [O_{p_{1,j}}(1 - O_{p_{2,j}}) + O_{p_{2,j}}(1 - O_{p_{1,j}})].
\]  
(3.2)

Note that the between-class criterion is minimized when the output activations for patterns belonging to different classes are different—that is, when $O_{p_{1,j}} \neq O_{p_{2,j}}$.

In a previous paper (Sarukkai 1994), I derived the following learning rule:

\[
\Delta W_{ij} = \sum_{\forall k, p \in \Omega_k} \frac{\partial O_{p_{1,j}}}{\partial W_{ij}} \left[ \eta_1 |\Omega_k||O_{p_{1,j}} - O_{p_{1,j}}| + \eta_2 |\Omega_k|[1 - 2O_{p_{1,j}}] \right].
\]  
(3.3)
where $\eta_1$ and $\eta_2$ are learning rates. $\overline{O_{\Theta_{\tilde{k}}}}$ is the thresholded, average output activation of unit $j$ for all patterns not belonging to class $\Omega_{\tilde{k}}$, and $|\Theta_{\tilde{k}}|$ refers to the cardinality of the set $\Theta_{\tilde{k}}$. $\overline{O_{\Omega_{\tilde{k}}}}$ is the thresholded, average activation of unit $j$ for class $\Omega_{\tilde{k}}$, and $|\Omega_{\tilde{k}}|$ refers to the cardinality of the set $\Omega_{\tilde{k}}$. Thresholded average activation is the threshold function applied to the average of the activations of a unit for input patterns belonging to the appropriate subset of the training data.

The advantage of my formulation is that gradient weight adaptation rules can be easily implemented in a neural network backpropagation context. The derived weight adaptation rule does have some intuitive explanations. In the first criterion, it can be seen that the average activation of a unit for its class acts as the teaching signal. From the second criterion, it may be seen that if the average activation of a unit for “other classes” is greater than 0.5, then the term $[1 - 2\overline{O_{\Theta_{\tilde{k}}}}]$ is negative (equivalent to a teaching signal of 0 for the “current” class), and vice versa. Thus, the equations clearly express the two criteria lucidly. The cardinality terms may be viewed as weighing constants related to the training set size.

In some cases, the within-class criterion and the between-class criterion are “opposing” quantities: if the class means for two classes are equal, then the effect of the between-class discernibility function is cancelled out by the within-class uniformity criterion. This problem can be alleviated by separating the training procedure into two phases, as shown in Table 1. The up arrow indicates that the value of the corresponding learning parameter is effectively higher, and the down arrow indicates that the value of the corresponding learning parameter is effectively lower during that learning phase. In the first phase, means of outputs for inputs belonging to the same class are computed. These class means are moved away from each other by applying the class discernibility phase of the algorithm, until the thresholded class means are orthogonal to each other. In practice, we find convergence. At this point, the within-class uniformity phase begins, setting a very low value for $\eta_2$ (typically zero, although a nonzero value ensures that the orthogonality of the output class labels is not immediately lost) and a higher value for $\eta_1$. Ideally, the system is allowed to switch back and forth between the two phases until orthogonal output class means with very low intraclass distances are achieved. Thus, both the functional mapping and the output class labels are iteratively developed at the same time. Since average activation values for the patterns are required, an additional batch forward pass may be required, although the averages computed from the previous pass can be used. In practice, it is better to do pairwise class discriminative training.

---

1 Without loss of generality, the activations are assumed to be bounded (0,1) in this particular formulation.
Table 1: Overview of the Algorithm

<table>
<thead>
<tr>
<th>Algorithm SOS: Self-organizing supervision with neural nets</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Class discernibility phase:</strong> Push between-class output cluster means away from each other until thresholded means for different classes are orthogonal.</td>
</tr>
<tr>
<td>$\eta_2 \uparrow, \eta_1 \downarrow$</td>
</tr>
<tr>
<td><strong>Class uniformity phase:</strong> Move within-class outputs toward class cluster means.</td>
</tr>
<tr>
<td>$\eta_2 \downarrow, \eta_1 \uparrow$</td>
</tr>
</tbody>
</table>

Thus, the learning algorithm performs gradient descent on the sum of the between-class discernibility and within-class uniformity criteria; however, only one criterion has a dominant effect on the learning phase at a given time. It can be analytically proved that the maximization of the interclass distance defines an MSE (mean squared error) linear discriminant function (Young & Calvert 1974). Thus, convergence can be guaranteed when the self-coding algorithm is applied to a linearly separable two-class problem using a single-layered perceptron. In other cases, once orthogonal class means are achieved and the self-organized codes stabilize, the method is equivalent to learning with fixed output labels; however, since the initial configuration of the system determines the output codes developed, it would be interesting to study whether the proposed scheme is less prone to local minima than traditional labeled supervision. The computational complexity of this batch algorithm is the same as traditionally labeled algorithms, if the required mean activations are computed from a previous epoch. For the experiments presented, orthogonality was generally achieved within a few hundred epochs, and convergence was achieved within a few thousand epochs.

4 Experiment I: Iris Data

The idea was to test the classification of the three-class iris data set using the self-organization process detailed earlier. The data set contains three classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other two, but the latter are not linearly separable from each other. The input attributes are sepal length, sepal width, petal length, and petal width. The network consisted of four inputs, four hidden units, and three output units. Sixty trials were performed with random initial conditions (weight range $[-3.7, 3.7]$). Training proceeded until the total squared error was less than 1.0 (one trial failed to converge). In the 59 successful trials, the supervised self-organization scheme achieved an average classification of 98.96% (maximum, 99.33%; minimum, 98.67%). Sammon’s stress (PCA, LDA, and NDA; results are from Mao & Jain 1993) measures how well the projection preserves the interpattern distances.
Table 2: Sammon’s Stress Obtained for the Iris Data Set

<table>
<thead>
<tr>
<th>Sammon’s stress</th>
<th>PCA</th>
<th>LDA</th>
<th>NDA</th>
<th>Self-organized</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>0.147</td>
<td>0.519</td>
<td>1.513</td>
<td>0.510</td>
</tr>
</tbody>
</table>

(i.e., input space versus output space).\(^2\) Sammon’s stress (Sammon 1969) is defined as

\[
E = \frac{1}{\sum_{\mu=1}^{n-1} \sum_{\nu=\mu+1}^{n} d^*(\mu, \nu)} \sum_{\mu=1}^{n-1} \sum_{\nu=\mu+1}^{n} \frac{(d^*(\mu, \nu) - d(\mu, \nu))^2}{d^*(\mu, \nu)}, \tag{4.1}
\]

where \(d^*(\mu, \nu)\) and \(d(\mu, \nu)\) are the (Euclidean) distances between pattern \(\mu\) and pattern \(\nu\) in the input space and in the projected space, respectively.

It can be seen from Table 2 that good values for Sammon’s stress have been obtained by the nonlinear self-organized projections, suggesting that the distances between patterns in the input space are captured, to some degree (comparable to an LDA network), in the self-organized output class projections.

5 Experiment II: Multispeaker Syllable Classification

In this experiment, described in more detail in Sarukkai (1994), four syllables were used: ba, pa, da, and ga. The total number of speaker-dependent speech tokens was 240; half were used for training and the other half for testing. Eighty speech tokens, from two speakers, which were not used in the training set, constituted the new speaker test set. A total of 24 Melscale filter bank coefficients were computed every 20 ms duration (window offset, 10 ms), and this constituted a total of 192 parameters, which were then linearly normalized. The network architecture consisted of 192 input units, one hidden layer of 50 units, and 6 output units. Although the data used in experiments II and III are fixed length, the algorithm can be applied in conjunction with dynamic input length architectures like time delay neural networks (TDNNs). The benchmark neural net was trained with 4-bit (one per class) and 2-bit (encoded) output codings. Euclidean distance was used for classification. Tables 3 and 4 show the results for the multispeaker syllable task for new speaker and trained speaker test sets (13 trials supervised self-organization; 15 trials labeled supervision).

\(^2\) LDA: linear discriminant analysis; NDA: nonlinear discriminant analysis; PCA: principal component analysis.
Table 3: Multispeaker Syllable Recognition Results (in % with Standard Deviations)

<table>
<thead>
<tr>
<th></th>
<th>New speakers</th>
<th>trained Speakers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Self-organizing backprop</td>
<td>67.69 ± 7.11</td>
<td>77.5 ± 2.75</td>
</tr>
<tr>
<td>Normal backprop (2-bit output)</td>
<td>65.25 ± 7.34</td>
<td>77.11 ± 4.25</td>
</tr>
<tr>
<td>Normal backprop (4-bit output)</td>
<td>69.17 ± 6.14</td>
<td>83.17 ± 3.19</td>
</tr>
</tbody>
</table>

Table 4: Some Output Labels Produced by the Class Separability Scheme (Experiment II)

<table>
<thead>
<tr>
<th>Token</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>ba</td>
<td>100100</td>
<td>010100</td>
<td>011110</td>
</tr>
<tr>
<td>pa</td>
<td>100110</td>
<td>110100</td>
<td>110110</td>
</tr>
<tr>
<td>da</td>
<td>010001</td>
<td>000011</td>
<td>000001</td>
</tr>
<tr>
<td>ga</td>
<td>010011</td>
<td>100011</td>
<td>100001</td>
</tr>
</tbody>
</table>

Note: The italics accentuate the differences in the output values.

6 Experiment III: Multimodal Data

The database for this data set, described in more detail in deSa (1994), consists of auditory and visual features recorded from many speakers repeating one of the syllables: ba, va, da, ga, or wa. Multimodal data obtained from four speakers were used. The acoustic data were low-pass filtered and segmented automatically (using the time-domain wave magnitude program available in ESPS software from Entropic Research Laboratory). In order to ensure that all the consonantal information was retained, 50 ms from before and after the detected utterance was segmented. These utterances were then encoded using a 24-channel mel code over 20 ms windows overlapped by 10 ms. This gave a 216-dimension auditory code for each utterance. The visual frames were digitized as 64 × 64 8-bit gray-level images using the Datacube MaxVideo system (see Figures 1 and 2). The segmentation obtained from the acoustic signal was used to segment the video six frames before acoustically determined utterance onset and four after. The normal flow was

Figure 1: Example of da utterance.
computed using differential techniques between successive frames. Finally, using averaging techniques, five frames of motion over the $64 \times 64$-pixel grid were obtained. The frames were then divided into 25 equal areas ($5 \times 5$) and the motion magnitudes within each frame averaged within each area. Thus, a final visual feature vector of dimension ($5 \times 25$) 125 was obtained. Each input pattern consisted of the concatenated 216-dimensional auditory channel vector and the 125-dimensional visual parameter vector. The total number of patterns were 500. They were divided arbitrarily into 350 training samples and 150 test patterns, so that all speakers were represented in the training data.

The network architecture used for this problem is shown in Figure 3. The benchmark was to train the network with two-per-class output coding. Results tabulated (see Table 5) are from 39 trials for the backpropagation with 2-per-class coding, and 42 trials for the self-organizing coding scheme. The
Table 5: Test Set Results on the Multimodal Data Using Different Methods (%)

<table>
<thead>
<tr>
<th>Method</th>
<th>VQ</th>
<th>LVQ2.1</th>
<th>M-D</th>
<th>BP with output</th>
<th>Self-organize</th>
</tr>
</thead>
</table>

Note: A: auditory data only. V: Visual data only. B: Both modalities used.

results of the self-organizing scheme compare favorably with the labeled supervision scheme. The mean results for the same data set using Kohonen’s supervised LVQ, unsupervised VQ, and the M-D algorithms are cited from de Sa (1994). For the LVQ and M-D tests, 30 codebook vectors were used for the auditory and 60 for the visual data. LVQ2.1 (which is supervised) has the best performance on the test sets, mainly because 30/60 codebook vectors seem sufficient to capture the 350 training samples well.

In fact, the self-organization process goes further than combining features from both modalities. The closeness of patterns in the input space is reflected in the output encodings that the supervised self-organization scheme develops. Figure 4 shows the statistics of the interclass Hamming distances for the five classes as measured by the codes developed by the self-organizing network over 42 trials. The closest output coding (based on interclass Hamming distance) is (da, ga). This is interesting since experiments (Miller 1955) indicate that the syllables da and ga are perceptually often confused. Da and ga are also visually extremely alike (see Figures 1 and 2). It must be noted that although da, ba, and ga are all voiced stops, ba can be distinguished from the other two easily from the visual modality. In general, the place of articulation can be seen easily, but it is the hardest feature to hear. The pair with the next lowest interclass Hamming distance is the (va, ba) pair. Va is a voiced front fricative and is visually similar to ba. Although the production of ba requires that the lips be closed before the voice release, this visual cue is absent in the motion flow parameters that are computed from the visual images and used as inputs for the visual modality. The point of Figure 4 is to suggest that the self-organized codes are not totally random and seem to be biased by factors such as interpattern relationships in the input space.

7 Conclusions and Future Directions

Supervised learning in conventional multilayered neural networks can be formulated without explicitly specifying output label functions. Learning rules were defined on the basis of between-class and within-class error criteria. The actual criteria could be extended to incorporate more general constraints such as error-correcting output labels. The proposed scheme has
Figure 4: Average interclass Hamming distances with standard deviations for the various class pairs developed by the self-organizing network.

...been experimentally tested on various data sets. Interesting features emerge in the output representations of such self-organizing networks: some of the “intrinsic” or “natural” distances are somewhat preserved in the output projections. As a proof of concept, it has been shown that networks can self-organize in a supervised manner and develop their own interesting output labels.

Appendix: Details of Experiments

The cardinality terms were factored into the learning rates.  
\( \alpha \) is the momentum factor. The activation function used is \( \frac{1}{1 + e^{-0.667x}} \). Sigmoid prime offset was 0.1. Weights initialized \([-3.7 : 3.7]\).

I. Iris Experiment

Number of classes: 3  
Network architecture: 4 inputs, 4 hidden, 3 outputs  
Class discernibility phase parameters: \( \eta_1 = 0.00; \eta_2 = 0.01; \alpha = 0.00 \)
Supervised Networks That Self-Organize Class Outputs

Class uniformity phase parameters: $\eta_1 = 0.20; \eta_2 = 0.00; \alpha = 0.8$

II. Multispeaker Syllable Experiment

Number of classes: 4
Network architecture: 192 inputs, 50 hidden, 6 outputs
Class discernibility phase parameters: $\eta_1 = 0.00; \eta_2 = 0.09; \alpha = 0.1$
Class uniformity phase parameters: $\eta_1 = 0.08; \eta_2 = 0.02; \alpha = 0.6$
Benchmark parameters: $\eta = 0.1; \alpha = 0.1$ initially; $\alpha = 0.6$ after 200 epochs

III. Multimodal Data Experiment

Number of classes: 5
Network architecture: 216 auditory plus 125 visual inputs, 40 hidden, 10 outputs
Class discernibility phase parameters: $\eta_1 = 0.00; \eta_2 = 0.01; \alpha = 0.00$
Class uniformity phase parameters: $\eta_1 = 0.20; \eta_2 = 0.00; \alpha = 0.8$
Benchmark parameters: $\eta = 0.20; \alpha = 0.8$

Acknowledgments

I thank Dana H. Ballard for his valuable comments on this work. I am also grateful to Anil K. Jain for his important suggestions and pointers to related work. Also, thanks to Virginia deSa for providing the multimodal data. Thanks to the anonymous reviewers for their constructive criticism and insightful comments. This material is based on work supported by the National Science Foundation under grant IRI-8903582. The government has certain rights in this material.

References


Received November 14, 1994; accepted May 6, 1996.