Competitive Learning with Feedforward Supervisory Signal for Pre-trained Multilayered Networks

Takashi Shinozaki and Yasushi Naruse
Brain Imaging Technology Laboratory, NICT
Kobe, Hyogo 651-2493, JAPAN
tshino@nict.go.jp

Abstract

We propose a novel learning method for multilayered neural networks which uses feedforward supervisory signal and associates the classification of a new input with that of pre-trained input. The proposed method effectively uses rich input information in the earlier layer for robust learning and revising of the internal representation in a multilayer neural network.

1 Introduction

A multilayered deep neural network is one of the most powerful methods for human-like recognition tasks, image [1] and speech recognition [2]. Some previous studies demonstrated great performances with supervised learning in signal classification tasks [3, 4]. Supervised learning for multilayered neural network generally uses gradient-based learning rules, in particular, back-propagation learning [5]. However, the amount of supervisory information in the last layer is insufficient to supervise the whole deep neural network because the information is selected and reduced from layer to layer. This tendency is more serious in signal classification tasks which have limited discrete output. Thus, layer-wised learning is generally used for both mutually connected [6] and feedforward networks, resulting in difficulties in incremental learning and online updating.

This study proposes a novel learning method for multilayered neural networks that uses feedforward supervisory signal and associates the classification of a new input with that of pre-trained input [7]. The proposed method effectively uses rich input information in the earlier layer for robust learning and revising of the internal representation in a multilayer neural network.

2 Methods

2.1 Network structure

The network structure is inspired by the human visual system in the cortex. The network is composed of self-organizing-map (SOM) modules [8]. One hundred neurons in a SOM module have the same receptive field (RF) which receives a part of the output of the previous layer. Fig.1(a) shows a schematic of the network structure, and Table 1 shows the parameter values. As exemplified, the first layer consists of 49 SOMs with 4,900 neurons, and neurons receive a 6×6-pixel image that is a part of the input image with 28×28 pixels, resulting in a total of 176,400 connections in the layer.

Each neuron calculates an inner product between the weight and the input, and each SOM module behaves as a Winners Share All (WSA): the neuron which has the most prominent value in the module outputs 1.0, and neighboring neurons output a distance-decayed value determined using the Gaussian kernel, \( G(d) = \exp(-d^2/2\sigma^2) \). Here, \( d \) is the spatial distance from the winning neuron; we set \( \sigma = 0.4 \).
**Table 1: Network parameters**

<table>
<thead>
<tr>
<th>Layer</th>
<th>Maps</th>
<th>Neurons</th>
<th>RF</th>
<th>RF stride</th>
<th>Connections</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7×7</td>
<td>4,900</td>
<td>6×6 pixels</td>
<td>4×4 pixels</td>
<td>176,400</td>
</tr>
<tr>
<td>2</td>
<td>3×3</td>
<td>900</td>
<td>5×5 maps</td>
<td>1×1 map</td>
<td>2,250,000</td>
</tr>
<tr>
<td>3</td>
<td>1×1</td>
<td>100</td>
<td>3×3 maps</td>
<td>1×1 map</td>
<td>90,000</td>
</tr>
</tbody>
</table>

2.2 Pre-training

We use traditional unsupervised competitive learning for pre-training [8, 9]. It updates the weight of the most prominent neuron and its neighbors, and forms a two-dimensional spatial structure of template sets for the input pattern. The update rule is described as follows,

\[ \Delta w = \alpha x_{\text{input}} \exp(-d^2/2\sigma^2) \]  

where \( w \) is the weight vector, \( x_{\text{input}} \) the input vector, and \( d \) the spatial distance from the winning neuron. The learning coefficient \( \alpha \) linearly decreases from 1.00 at the beginning to 0.00 at the end, and the standard deviation of the Gaussian kernel \( \sigma \) also decreases from 3.5 to 0.0. The weight vector is normalized at every update.

\[ w_{\text{new}} = (w + \Delta w)/|w + \Delta w| \]  

The learning method generates a spatially continuous feature map which is similar to the map generated using Topographic Independent Component Analysis (TICA) [10].

MNIST handwritten image dataset [11] (10 digits, 28×28 pixels, grayscale) is employed in pre-training and experiments. The pre-training were performed in a layer-wise manner, which is similar to a biological critical period. Initially, only the first layer is learned with 10,000 iterations. Next, 10,000 iterations are applied to the first and second layer. Finally, the first three layers are processed applying sequentially 40,000, 30,000 and 20,000 iterations. Fig.1(b) represents a typical example of generated feature maps located centrally in the first layer.

2.3 Advance Propagation learning

Advance propagation (AP) learning is a supervised learning method that combines two independent input signals. It is based on Learning Vector Quantization (LVQ) [12], but requires additional advance input as a supervisory signal. Before the target input, advance input, which produces the required classification label, is propagated through the whole network, and then the target input is processed with the after-effect of the advance input. LVQ-like conditional learning followed by the target input revises the weight vector to produce the required label.

A learning trial is processed as following: Firstly, the target input \( x_{\text{target}} \) is processed by the network, and the output label of the \( x_{\text{target}} \) is checked. If it is not the required label, the advance propagation learning is evoked. The advance input \( x_{\text{adv}} \) which produces the required label output is processed by the same network, resulting the required label output. Subsequently, the target input \( x_{\text{target}} \) is processed again with the after-effect as follows,

\[ x_{\text{input}} = \beta x_{\text{adv}} + (1 - \beta) x_{\text{target}} \]  

where \( \beta \) the ratio of the after-effect of the advance input. The vector represents the direction of the feature vector \( x_{\text{target}} \) corrected by the after-effect of the advance input \( x_{\text{adv}} \). The important point is that the network has highly non-linear behavior by WSA and the output is not equal to that produced by the linear summation of two inputs. The following competitive learning uses the combined input as same way as Eq.1. Consequently, the full version of the equation with a multi-layer decay and a Gaussian kernel for WSA output, described as follows,

\[ \Delta w = r^{n-l} \alpha (\beta x_{\text{adv}} + (1 - \beta) x_{\text{target}}) \exp(-d^2/2\sigma^2) \]  

where \( r \) the decay coefficient from layer to layer, \( n \) the total number of layers, and \( l \) the target layer. The weight vector is normalized at every update as in usual competitive learning (Eq.2). We used \((\alpha, \beta, r, n) = (0.05, 0.6, 0.5, 4)\) in experiments.
Figure 1: (a) A schematic of the network structure. (b) A typical example of the feature map in the first layer generated by unsupervised pre-training. (c) The error rate of the discrimination with AP learning among training set iterations. The inset is the detailed result of the initial part. Error bars are standard deviations (n=10). (d) The usage of neurons in the last layer. Ten representative neurons remained after AP learning. (e) Optimal stimuli of a typical representative neuron (which codes ‘6’) before and after AP learning.

Table 2: Comparison of the error rate

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Test error rate [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed learning method</td>
<td>4.2</td>
</tr>
<tr>
<td>2-layer NN, 1000 hidden units (LeCun et al., 1998) [13]</td>
<td>4.5</td>
</tr>
<tr>
<td>Convolutional net LeNet-5 (LeCun et al., 1998) [13]</td>
<td>0.95</td>
</tr>
<tr>
<td>Committee of 35 conv. net (Ciresan et al., 2012) [14]</td>
<td>0.23</td>
</tr>
</tbody>
</table>

3 Experiments

AP learning is applied to the pre-trained network. The most matched output of the last layer in the pre-training result is selected for each label. Advance inputs as supervisory signals are dynamically determined, and updated from one trial to the next. Each input signal is initially tested by its label, and AP learning is applied if the label is incorrect. The block size of the iteration is 10,000, and the error rate is checked at each end of the block.

The error rate firstly increased from the result of the pre-training, and then decreased stably in following learning blocks (Fig 1(c)). AP learning improved the error rate from 15.6±0.7 % just after the pre-training, to 4.2±0.2 % after 20 iterations of the training set (n=10). The initial increase of the error rate might be caused by a re-arrangement of the internal representation: the initial scattered representation was re-arranged to more sparse and efficient style (Fig 1(d)). At the same time, optimal stimuli of representative neurons are also modified to more generalized image (Fig 1(e)).

4 Conclusion

In this paper, we introduced a novel supervised learning for a deep feedforward neural network, and validated the efficiency in the visual recognition task. The proposed learning method focuses to use the rich input information in the early layer as the supervisory signal in each layer. The result exhibits the proposed method could work supervised fine-tuning on the pre-trained multilayered network (Fig 1(c)), and reorganizes the representation in the network drastically (Fig 1(d)). However, the proposed method is still unmatured, and is inferior to results of previous reports [13, 14]. Table 2 shows the comparison with some previous results of MNIST task. The result indicates the
The performance of the proposed method is in the primitive level of other learning methods. In the future study, we will try to improve the performance by more optimized parameters of the learning and the network structure, and by a refined version of the proposed learning method with the local activity dependent update rule which could enable more spatially selective fine-tuning of the weight vector.

The proposed learning method is applied to the whole network concurrently, and not layer by layer. Only the correct/incorrect signal is broadcasted among the network, and each local module uses the broadcasted signal and locally propagated information, enabling easy implementation to recent highly parallel computing systems. We are planning to implement the learning method on GPGPUs in the future study.

One of the interesting points the proposed learning method is that it could seamlessly incorporate both reinforcement learning and competitive learning. Reinforcement learning emerges if there is no advance input, and the usual competitive learning emerges if there is no correct/incorrect signal. This suggests that these learning methods can share the same hardware implementation, and the learning mode can be selected by the sequence of input and correct/incorrect signals. Moreover, the timing of the correct/incorrect signal can control the associative layer, and it might be useful for deeper or recurrent networks.

AP learning associates a new input image to a pre-trained label. It suggests that new natural images can be associated with labels that are pre-trained by clean artificial images. On the other hand, the proposed method could be extended to multiple advance inputs (for example, use ‘red’ and ‘round shape’ to learn ‘apple’). Thus, the learning method could be applicable for letter classification in natural images. Further studies are required.

References


